НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
ФІЗИКО-ТЕХНІЧНИЙ ІНСТИТУТ
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ ЗАХИСТУ ІНФОРМАЦІЇ

«До захисту допущено»
В.о. завідувача кафедрою
_____ М.М.Савчук
(підпис) (ініціали, прізвище)
“___”_____20 _ р.

Дипломна робота

на здобуття ступеня бакалавра

з напряму підготовки : 113 «Прикладна математика»
(код і назва)

на тему: Оцінка кількості вентилів для реалізації шифру Калина в квантовій моделі обчислень

Виконав (-ла): студент (-ка) 4 курсу, групи ФІ-62
(шифр групи)

Салій Роман Васильович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник к.т.н Фесенко А.В. _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020року

Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»
Фізико-технічний інститут

Кафедра математичних методів захисту інформації

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки - 113 «Прикладна математика»

ЗАТВЕРДЖУЮ
В.о. завідувача кафедрою

М.М.Савчук

_____
(підпис) (ініціали, прізвище)

«___»_____20__ р.

ЗАВДАННЯ
на дипломну роботу студенту

_____
(прізвище, ім'я, по батькові)

1. Тема роботи Оцінка кількості квантових вентилів для реалізації шифру Калина в квантовій моделі обчислень _____,
керівник роботи к.т.н. Фесенко А.В. _____ ,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від _____ р. № _____

2. Термін подання студентом роботи _____

3. Вихідні дані до роботи роботи, що стосуються квантових вентилів, алгоритму Гровера та шифру Калина _____

4. Зміст роботи у цій роботі було проведено аналіз оцінки, котра була отримана для шифру AES, дослідження шифру Калина та оцінка кількості вентилів для його реалізації в квантовій моделі обчислень _____
_____
_____
_____

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) презентація _____
_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Огляд опублікованих наукових праць за темою дослідження		
2	Аналіз оцінки отриманої для шифру AES		
3	Дослідження шифру Калина		
4	Оцінка кількості вентилів для шифру Калина		

Студент _____ ___Салій Р.В_____
(підпис) (ініціали, прізвище)

Керівник роботи _____ ____Фесенко А.В.____
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Кваліфікаційна робота містить: 94 сторінок, 12 рисунків, 22 таблиці, 31 джерело, 1 додаток.

Мета роботи: Визначення кількості необхідних ресурсів, таких як квантові вентилі, на реалізацію шифру Калина.

Об'єкт дослідження: Інформаційні процеси в системах криптографічного захисту в квантовій моделі обчислень.

Предмет дослідження: Складність реалізації шифру Калина в квантовій моделі обчислень.

У результаті цієї роботи було отримано оцінку кількості вентилів Тоффолі, необхідних для реалізації шифру Калина в квантовій моделі обчислень. Спочатку було оцінено кількості вентилів, необхідних для виконання підстановок, операції AddRoundKey та раундів шифру. Отримано, що для одного виконання операції AddRoundKey необхідно близько 12.737 вентилів Тоффолі, а для одного раунду шифрування — близько 17.832 вентилів Тоффолі. Для реалізації підстановок шифру Калина нуобхідно 1278, 1273, 1252 та 1262 вентилів Тоффолі, для π_0, π_1, π_2 та π_3 відповідно. А для повної реалізації шифру Калина-k/k необхідно близько 202.600, 273.510 та 344.420 вентилів Тоффолі, для $k = 128, 256, 512$ відповідно.

ШИФР КАЛИНА, КВАНТОВА МОДЕЛЬ ОБЧИСЛЕНЬ, КВАНТОВІ ВЕНТИЛІ

РЕФЕРАТ

Квалификационная робота содержит: 94 страниц, 12 рисунков, 22 таблицы, 31 источник, 1 приложение.

Цель работы: Определение количества необходимых ресурсов, таких как квантовые вентили, для реализации шифра Калина.

Объект исследования: Информационные процессы в системах криптографической защиты в квантовой модели вычислений.

Предмет исследования: Сложность реализации шифра Калина в квантовой модели вычислений.

В результате этой работы было получено оценку количества вентилей Тоффоли, необходимых для реализации шифра Калина в квантовой модели вычислений. Сначала было оценено количества вентилей, необходимых для выпосления подстановок, операции AddRoundKey та раундов шифра. Получено, что для одного выполнения операции AddRoundKey необходимо 12.737 вентилей Тоффоли, а для одного раунда шифровки — около 17.832 вентилей Тоффоли. Для реализации подстановок шифра Калина необходимо 1278, 1273, 1252 та 1262 вентилей Тоффоли, для π_0, π_1, π_2 та π_3 соответственно. А для полной реализации шифра Калина-k/k нужно около 202.600, 273.510 и 344.420 вентилей Тоффоли, для $k = 128, 256, 512$ соответственно.

ШИФР КАЛИНА, КВАНТОВАЯ МОДЕЛЬ ВЫЧИСЛЕНИЙ, КВАНТОВЫЕ ВЕНТИЛИ

ABSTRACT

The work contains 94 pages, 12 illustrations, 22 tables, 1 appendices, 31 sources of literature.

The aim of work: Estimation of the amount of necessary resources, such as quantum gates, for the implementation of the Kalyna cipher.

Object of research: Information processes in cryptographic protection systems in a quantum computing model.

Subject of research: The complexity of the implementation of the Kalyna cipher in the quantum model of computing.

As a result of this work, an estimate was obtained of the number of Toffoli gates needed to implement the Kalyna cipher in the quantum computing model. First, the number of gates needed to perform the substitutions, the AddRoundKey operation, and the cipher rounds was estimated. It was found that for one execution of the AddRoundKey operation, 12.737 Toffoli gates are needed, and for one round of encryption, about 17.832 Toffoli gates. To implement Kalyna cipher permutations, needed 1278, 1273, 1252 та 1262 Toffoli gates, for π_0, π_1, π_2 and π_3 And for the full implementation of the Kalyna-k/k cipher about 202.600, 273.510 and 344.420 Toffoli gates are needed, for $k = 128, 256, 512$.

KALYNA CIPHER, QUANTUM COMPUTING MODEL, QUANTUM GATES

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Вентиль NOT — виконує над кубітом логічну операцію заперечення.

Вентиль Toffoli(CCNOT) — універсальний контрольований оборотний вентиль з трьома входами.

Вентиль CNOT — оборотний вентиль, має два входи і два виходи. Перший вхід - біт, який буде інвертовано в випадку, коли на другий буде подано одиницю.

ВСТУП

Актуальність дослідження. Виникнення квантових комп'ютерів і квантових алгоритмів різко змінило спільноту криптографів. Найбільш відомі криптографії з відкритим ключем, RSA та криптографії еліптичних кривих, засновані на труднощах задачі факторизації та дискретної алгебри. Проте є алгоритми, котрі допомагають легко вирішувати арифметичні задачі, роблячи RSA і ECC вразливими.

В області симетричних ключів, вплив квантових комп'ютерів не дуже критичний, як в випадку з відкритими ключами. Алгоритм Гровера може бути використаний для пошуку n-бітого секретного ключа зі швидкістю n, це найбільш ефективний метод квантової атаки на блоковий шифр. Застосування алгоритму Гровера до блочкових шифрів є найкращим способом вимірювання рівня їх захищеності від атак зі сторони квантових комп'ютерів. По цій причині не тільки алгоритм Гровера, а й криптографія, котру необхідно проаналізувати повинна бути реалізована за допомогою квантової моделі. Оскільки розробка квантового комп'ютеру знаходиться в рудиментарному стані, пошук оптимального квантового ресурсу для цільового алгоритму являється одним з найбільш важливих питань.

Вже було оцінено кількість квантових ресурсів, необхідних для реалізації деяких криптографічних алгоритмів в квантовій моделі обчислень. В цій роботі буде проведено оцінку кількості необхідних квантових вентилів при реалізації вітчизняного шифру Калина в квантовій моделі обчислень.

Метою дослідження є визначення необхідних квантових ресурсів, таких як вентилі, на реалізацію шифру Калина. Для досягнення мети необхідно виконати такі задачі дослідження:

1) Провести огляд існуючих методів оцінки необхідних квантових вентилів при реалізації криптографічних примітивів;

2) Проаналізувати існуючі оцінки складності реалізації криптографічних алгоритмів;

3) Дослідити особливості роботи шифру Калина національного стандарту ДСТУ 7624:2014;

4) Провести оцінку кількості квантових вентилів окремих складових шифру Калина в квантовій моделі обчислень для обчислення загальної кількості вентилів.

Об'єктом дослідження є інформаційні процеси в системах криптографічного захисту в квантовій моделі обчислень.

Предметом дослідження є складність реалізації шифру Калина в квантовій моделі обчислень.

При розв'язанні поставлених завдань використовувались такі *методи дослідження*: методи лінійної та абстрактної алгебри, теорії імовірностей, теорії складності алгоритмів.

Наукова новизна отриманих результатів полягає у тому що вперше було обчислено оцінку кількості необхідних вентилів для реалізації шифру Калина в квантовій моделі обчислень. Також, додатково отримано мінімальні нормальні форми булевих формул, котрі відповідають підстановкам шифру Калина.

Практичне значення результатів полягає в тому, що було отримано мінімальну нормальну форму для блоків підстановок, котрі можна використати при побудові атак на шифр Калина. Отримана оцінка складності реалізації шифру Калина дозволяє оцінити можливість використання цього шифру на існуючих екземплярах квантового комп'ютеру.

1 ОГЛЯД ІСНУЮЧИХ ТЕОРЕТИЧНИХ МАТЕРІАЛІВ ЗА ТЕМОЮ ДОСЛІДЖЕННЯ

В даному розділі буде проаналізовано існуючі наукові джерела, в яких описані теоретичні дані, котрі необхідні для виконання поставленої задачі. Наведено визначення шифрів AES та Калина, атаки на ці шифри та опис алгориму Гровера.

1.1 Аналіз останніх досліджень і публікацій

В Україні з 1990-го року в якості блокового симетричного шифру використовувався ДСТУ 28147:89 з розміром блоку 64 біти та розміром ключа 256 біт [1]. Для нього вже давно відомі теоретичні методи криптоаналізу, за складністю сильно меншою, ніж повний перебір ключів [2]. Швидкий розвиток IT-сфери та збільшення обсягів інформації призвели до того, що довжини блоку в 64 біти вже недостатньо для роботи з великими обсягами інформації.

ГОСТ 28147-89 значно поступається сучасшим аналогам, таким як AES, але заміна ГОСТ 28147-89 на AES не мала сенсу, тому що світові тенденції свідчили про початок відмови від цього стандарту. Лідери IT-індустрії вже давно відмовилися від використання AES, та застосовують нові алгоритми замість нього.

З 01.07.2015 року в Україні було введено в дію ніціональний криптографічний стандарт ДСТУ 7624:2014, котрий визначає шифр "Калина". Новий стандарт підтримує 128, 256 та 512 бітні ключі, забезпечуючи гарний рівень стійкості. ДСТУ 7624:2014 єдиний в світі стандарт, котрий підтримує 512-бітні симетричні ключі.

Розглянемо вже проведені розрахунки(Таблиця 1.1) ресурсоємності БСШ, котрі приведені в [3]. Було проведено розрахунок необхідної

кількості процесорних інструкцій на обробку одного байту даних. Обчислення проводились на платформі 64-бітового мікропроцесора з архітектурою x86-64 від фірми Intel.

Таблиця 1.1 – Кількість раундів і кількість рядків у матриці стану для різних значень розміру блоку та довжини ключа.

	Блоковий симетричний шифр	
	Калина(128/128)	AES
К-сть операцій на 1 байт	40.375	45.375

Для найбільшої швидкодії було обрану мову програмування C++, використовувася компілятор gcc version 4.9.2, комп'ютер під управлінням ОС Linux(64 біти) з процесором Intel Core i5-4670(3.40 ГГц). При реалізації AES інструкції AES-NI не використовувались. Результати тестування швидкості реалізації (при максимальній оптимізації компілятора) [4] наведено на Рис. 1.1

Автори отримали наступні висновки(при використанні 64-бітової платформи):

–швидкість Калини вища за швидкість AES на 3% (86 Мбіт/с) для ключа довжини 128 біт;

–швидкість Калини нижча за швидкість AES-128 на 10% для ключа довжини 256 біт;

–швидкість Калини вища за швидкість AES-256 на 1% для ключа довжини 256 біт;

–швидкість Калини в 2.8 рази більша за швидкість ГОСТ 28147-89 з довжиною блоку 128 біт;

–швидкість Калини в 3.16 рази більша за швидкість ГОСТ 28147-89 з довжиною блоку 256 біт;

–швидкість Калини приблизно в два рази більша за швидкості нових стандартів шифрування Білорусії та Російської Федерації.

Результати, отримані Совиним та ін. [5] показали, що для

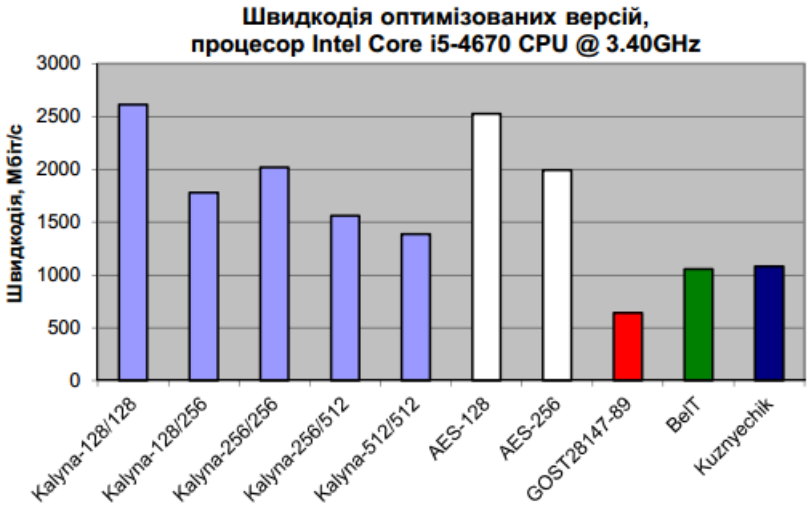

Рисунок 1.1 – Швидкодія блокових шифрів.

вбудованих систем з достатьо нормальними і високими рівнями стійкості, Калина поступається AES і ГОСТ 28147-89 як швидкістю, так і кількістю необхідної пам'яті. Фактично, було підтверджено те, що Калина орієнтована на 64-бітні високопродуктивні мікропроцесори загального призначення, але не на системи які мають обмежені ресурси.

ДСТУ 7624:2014 розроблено Приватним акціонерним товариством «Інститут інформаційних технологій». Розробники: І. Горбенко, д-р техн. наук (науковий керівник); Ю. Горбенко, канд. техн. наук; О. Дирда, канд. техн. наук; В. Долгов, д-р техн. наук; Д. Кайдалов; О. Казимиров, канд. техн. наук; О. Кузнецов, д-р техн. наук; Р. Мордвінов; Р. Олійников, д-р техн. наук; А. Пушкарьов; В. Руженцев, канд. техн. наук

Цей стандарт установлює криптографічний алгоритм симетричного блокового перетворення для забезпечення конфіденційності та цілісності інформації під час її обробки. Стандарт використовують під час розроблення засобів криптографічного захисту інформації в інформаційних, телекомунікаційних та інформаційно-телекомунікаційних системах, а також під час модернізації діючих систем для заміни ГОСТ 28147-89.

1.2 Опис шифру Калина

Блоковий шифр "Калина"був обраний під час Українського національного публічного криптографічного конкурсу 2007-2010 років.

Шифр Калина має кілька режимів роботи, такі як:

1) Проста заміна(Базове перетворення).(Позн. ECB)

2) Гамування.(Позн. CTR)

3) Гамування зі зворотним зв'язком за шифруванням.(Позн. CFB)

4) Вироблення імітовставки.(Позн. CMAC)

5) Зчеплення шифроблоків.(Позн. CBC)

6) Гамування зі зворотним зв'язком за шифрограмою.(Позн. OFB)

7) Вибіркове гамування із прискореним виробленням імітовставки.(Позн. GCM/GMAC)

8) Вироблення імітовставки і гамування.(Позн. CCM)

9) Індексована заміна.(Позн. XTS)

10) Захист ключових даних.(Позн. KW)

Режим роботи криптографічного алгоритму, визначеного в ДСТУ 7624:2014, позначають так: «Калина-l/k-позначення режиму/параметри режиму» (для деяких режимів параметрів немає), де l - розмір блока базового перетворення, k - довжина ключа.

Наприклад, Калина-256/512-CCM-32, 128 визначає використання базового перетворення з розміром блока 256 бітів, довжиною ключа 512 бітів, застосування у режимі вироблення імітовставки і гамування, довжина конфіденційної (та відкритої) частини повідомлення завжди менше ніж 2^{32} байтів, довжина імітовставки дорівнює 128 бітів.

Шифр Калина-l/k-ECB-x

Далі ми будемо розглядати режим простої заміни (базове перетворення).

Загальні відомості. Шифр описано в [6]. Калина підтримує такі розміри блоків i довжини ключів, як 128, 256 і 512 біт. Довжина ключа

може дорівнювати або бути вдвічі більше за розмір блоку.

Базове перетворення зашифрування $T_{ij}^{(K)}$ є параметризованим ключем шифрування K відображенням $T_{ij}^{(K)}$: $V_l \to V_l$, $K \in V_k$, l,$k \in$ {128,256,512}, при цьому $k = l$ або $k = 2l$, що реалізовано у вигляді ітеративного застосування низки функцій, які обробляють вхідний аргумент $x \in V_l$, як матрицю внутрішнього стану розміром $8 \times c$ байтів, що містить елементи поля $GF(2^8)$.

Залежність кількості раундів (t) в разі реалізації перетворень $T_{ij}^{(K)}$ та $U_{ij}^{(K)}$, кількості стовпців матриці внутрішнього стану (c) від розміру блока і довжини ключа шифрування (k) наведено в Таблиці 1.2.

Таблиця 1.2 – Кількість раундів і кількість рядків у матриці стану для різних значень розміру блоку та довжини ключа.

	Розмір блоку(l)	Довжина ключа(k)	К-сть раундів(t)	К-сть рядків матриці(c)
1	128	128	10	2
2		256	14	
3	256	256	14	4
4		512	18	
5	512	512	18	8

Проста заміна оброблює вхідний блок данних довжиною l бітів(відкритого тексту або шифротексту). Матриця внутрішнього стану позначається як $G = (g_{ij})$, $g_{ij} \in GF(2^8)$, де $i \in [0;7]$, $j \in [0;c-1]$. Запис байтів до матриць здійснюється по стовпцях.

Шифрування

Базове перетворення $T_{ij}^{(K)}$ виглядає так:

$$T_{lk}^{(K)} = \eta_l^{K_t} \circ \psi_l \circ \tau_l \circ \pi_l' \circ (\prod_{v=1}^{t-1}(\kappa_l^{K_v} \circ \psi_l \circ \tau_l \circ \pi_l')) \circ \eta_l^{K_0}, \text{ де}$$

l - розмір внутрішнього стану блокового шифру (в бітах),

K - ключ шифрування,

k - довжина ключа шифрування в бітах,

$\eta_l^{K_t}$ - функція додавання циклового ключа K_v $(v \in \{0,t\})$ за $\mathrm{mod}2^{64}$

π'_l - шар нелінійного бієктивного відображення, який виконує оброблення векторів, заданих над V_8 (байтова підстановка),

τ_l - перестановка елементів $g_{ij} \in GF(2^8)$ внутрішнього стану (циклічний зсув рядків вправо),

ψ_l - лінійне перетворення (множення матриці лінійного перетворення на матрицю внутрішнього стану над скінченним полем),

$\kappa_l^{K_v}$ - інволютивне перетворення, тобто функція додавання циклового ключа K_v за модулем 2 ($v \in \{1,2,...,t\text{-}1\}$).

У функціях π'_l, τ_l і ψ_l вхідний елемент $x \in V_l$ та вихідне значення $\chi(x) \in \{\pi'_l,\ \tau_l,\ \psi_l\}$, при цьому вхідні аргументи та вихідне значення розглядають як матриці розміром $8 \times c$ байтів.

Функція додавання циклового ключа K_v ($v \in \{0,t\}$) за модулем 2^{64} ($\eta_l^{K_t}$) здійснює додавання за $\mathrm{mod}2^{64}$ стовпців матриці внутрішнього стану $G = (g_{ij})$ і стовпців матриці циклового ключа $K = (k_{ij})$, результатом буде внутрішній стан після додавання – матриця розміром $8 \times c$. У разі виконання додавання менші значущі байти мають менші індекси, тобто використовується формат little-endian.

Шар нелінійного бієктивного відображення (π'_l) – виконує заміну кожного елемента g_{ij} матриці внутрішнього стану $G = (g_{ij})$ на $\pi_{imod4}(g_{ij})$, де $\pi_s{:}V_8 \to V_8$, $s \in \{0,1,2,3\}$.

Перестановка елементів (τ_l) – виконує циклічний зсув вправо рядків матриці $G = (g_{ij})$. Кількість елементів зсуву залежить від номера рядка та розміру блока, обчислюється за формулою:

$$\delta_i = \lfloor \tfrac{i \cdot l}{512} \rfloor$$

Під час обчислення результату функції лінійного перетворення (ψ_l) кожен елемент g_{ij} матриці внутрішнього стану $G = (g_{ij})$ розглядають як елемент скінченного поля $GF(2^8)$, утвореного незвідним поліномом $\vartheta(x) = x^8 + x^4 + x^3 + x^2 + 1$, або $0 \times 11d$ у шіснадцятковому поданні.

Кожен елемент результуючої матриці стану $W = (w_{ij})$ отримують як результат множення векторів довжини 8 над скінченним полем $GF(2^8)$ за формулою: $w_{ij} = (\upsilon \ggg i) \otimes G_j$, де υ — вектор, що утворює

циркулянтну матрицю МДР-коду і складається з послідовності байтових констант у шістнадцятковому представленні, які інтерпретують як елементи поля $GF(2^8)$, при цьому циклічний зсув виконується відносно елементів вектора над скінченним полем, G_j — j-тий стовпець матриці стану $G = (g_{ij})$.

Функція додавання циклового ключа $(\eta_l^{K_t})$ має вхідний аргумент $x \in V_l$(внутрішній стан шифру) і залежить від параметра $K_v \in V_l$(циклового ключа v-ї ітерації), кожен з яких подано як матрицю розміром $8 \times c$ байтів.

1.3 Атаки на шифр Калина

В [7] було проведено *Meet-in-the-Middle* атаку на шифр Калина зі зменшеною кількістю раундів. Атака була націлена на 7-раундову Калину, де розмір ключа вдвічі більше розміру блока. Згідно з аналізом безпеки, котрий виконали розробники шифру, Калина стійка до різних криптограналітичних методів після 5-го та 6-го раундів 128-бітної та 256-бітної версій відповідно. В атаці автори будують атаку так, щоб уникнути ефекту розповсюдження носіїв після додавання модульного ключа, що покращує ймовірність правильного шляху, відповідно, зменшуючи складності данних та часу. Через неможливість відновлення основного ключа з раундових, було використано лінійне відношення між парними та непарними індексами раундових ключів, щоб можна було ефективно їх всі відновити.

В [8] було описано деякі атаки для запуску відновлення ключів Калини-128/256 і Калини-256/512. Було покращено 7-раундову атаку[30], для демонстрації перших 9-раундових атак. Ці атаки були кращі з точку зору часу та складності данних. Зараз ці атаки працюють для Калини-b/2b, а варіанти, коли розміри блоку та ключа рівні здаються безпечними. Результати вказують на те, що при порівнянні Калини-b/2b та Калини-b/b, друга виглядає більш стійкою.

1.4 Опис AES.

AES[9] розроблений Йоаном Дамен та Вінсентом Реймен, є чинним стандартом шифрування, вибраним на відкритому конкурсі, котрий організували NIST в 2000 році. Це мережа підстановок-перестановок, що чергується між лінійними шарами та доповненнями раундових ключів. Має три різних довжини ключів: 128, 192 та 256 біт з різними розкладами ключів, та 10, 12 і 14 раундами відповідно.

Розглянемо AES-128. Шифр шифрує блоки розміру 128 біт, розбитих на 16 байтів, організованих в вигляді квадрату (табл. 1.3). Раундова функція має чотири операції, AddRoundKey (ARK), котра ксорить раундовий ключ з поточним станом, SubBytes (SB), котра застосовує S-блок AES до кожного байту, ShiftRows (SR), котра зміщує i-тий ряд на i байтів наліво і MixColumns (MC), який множить кожен стовпчик на матрицю AES MDS.

Ключ шифру складається з 128 бітів, поділених на 16 байтів k_0,k_1,...,k_{15}, котрі записуються в стовпці матриці $InputKey$. Кожен стовпчик цієї матриці утворює слово, тобто, ключ шифру це чотири слова w_0,w_1,w_2,w_3, де $w_0 = k_0k_1k_2k_3$, $w_1 = k_4k_5k_6k_7$ і т.д.

Таблиця 1.3 – Представлення «розбитих» блоків тексту.

k_0	k_4	k_8	k_{12}
k_1	k_5	k_9	k_{13}
k_2	k_6	k_{10}	k_{14}
k_3	k_7	k_{11}	k_{15}

З цих слів, за допомогою спеціального алгоритму (який буде описано далі) формується послідовність з 44 слів: w_0,w_1,w_2,...,w_{43} (кожне з них по 32 біти). На кожен раунд подається по чотири слова з цієї послідовності, $w_0w_1w_2w_3$ — ключ першого раунду, $w_4w_5w_6w_7$ — ключ другого раунду і т.д.

Перед першим раундом виконується операція $AddRoundKey$ (додавання за модулем 2 з початковим ключем). Перетворення виконані в одному раунді позначають як $Round(State,\ RoundKey)$, де $State$ - матриця, котра описує дані на вході раунду і на його виході після шифрування; $RoundKey$ - матриця раундового ключа.

Раунд складається з чотирьох перетворень:

•$SubDytes$ - побайтова підстановка в S-блоці з фіксованою таблицею перестановок;

•$ShiftRows$ - побайтовий здвиг рядків матриці $State$ на різну кількість байт;

•$MixColumns$ - перемішування байт в стовпчиках;

•$AddRoundKey$ - XOR з раундовим ключем.

В останньому раунді $MixColumns$ немає.

Математичні основи шифру

Операції в полі $GF(2^8)$. Для опису алгоритму використовуеється кінцеве поле $GF(2^8)$ побудоване як розширення поля $GF(2) = \{0,1\}$ за модулем непривідного поліному $m(x) = x^8 + x^4 + x^3 + x + 1$. Елементами поля $GF(2^8)$ є елементи виду

$$b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0$$

степінь яких менше 8, а коефіцієнти $b_7,b_6,...,b_0 \in\{0,1\}$. Операції в полі виконуються за модулем $m(x)$. Всього в полі $GF(2^8)$ 256 многочленів. Представлення двійкового числа $b_7b_6b_5b_4b_3b_2b_1b_0$ в вигляді многочлену з коефіцієнтами $b_7,b_6,...,b_0$ дозволяє інтерпретувати байт як бітовий многочлен в полі $GF(2^8)$:

$$b(x) = b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0$$

Наприклад, байт 63 задає послідовність бітів 01100011 і позначає конкретний елемент поля

$$01100011 \leftrightarrow x^6 + x^5 + x + 1$$

Розгланемо основні математичні операції в полі $GF(2^8)$.

1) **Додавання байтів** виконується наступним чином: представити в вигляді бітових многочленів і додати по звичайному правилу додавання многочленів з приведенням коефіцієнтів суми за модулем 2.

2) **Множення байтів** виконується за допомогою представлення їх многочленами і множення за звичайними математичними правилами. Отриманий результат необхідно привести за модулем многочлена $m(x) = x^8 + x^4 + x^3 + x + 1$.

3) Для будь-якого ненульового многочлена $b(x)$ в полі $GF(2^8)$ існує **обернений** до нього по множенню $(b(x)b(x)^{-1} = 1 \mathrm{mod} m(x))$. Щоб знайти обернений елемент використовується розширений алгоритм Евкліда.

Раундове перетворення AES

Розглянемо більш детально перетворення раунду шифрування.

1. Операція *SubBytes*.

Операція виконує нелінійну заміну байтів, виконувану незалежно з кожним байтом матриці *State*. Заміна оборотна і побудована шляхом комбінації двох перетвореннь над вхідним байтом:

• знаходження оберненого елементу в полі $GF(2^8)$ (нульовий байт 00 переходить сам в себе).

• афінне перетворення: множення оберненого байту на многочлен $a(x) = x^4 + x^3 + x^2 + x + 1$ і додавання з многочленом $b(x) = x^6 + x^5 + x + 1$ в полі $F_2[x]/x^8 + 1$.

В матричній формі процедура *SubBytes* виглядає так:

$$\begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix}^{-1} + \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix},$$

де через x позначено вхідні біти, а через y - виходи. Якщо на вхід функції потрапляє нульовий біт, то результатом заміни будет число $y = b$. Процес заміни байтів за допомогою таблиці підстановок показано на рисунку 1.2. Нелінійність перетворення обумовлена нелінійністю інверсії x^{-1}, а оборотність - оборотністю матриці.

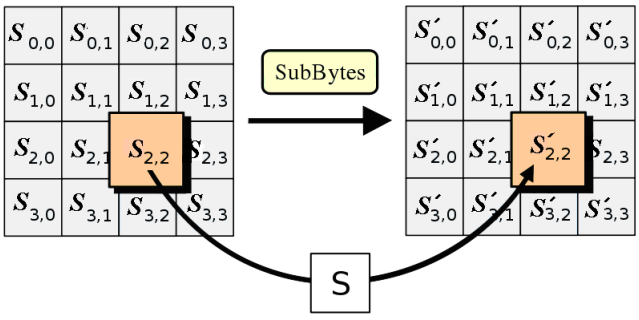

Рисунок 1.2 – Процес заміни байтів за допомогою таблиці підстановок.

Створену на основі даної операції таблицю перестановок байтів в шіснадцятковій системі називають **S-блоком**.

2. Операція *ShiftRows*.

Операція застосовується до рядків матриці *State* - перший не рухається, а нижні три циклічно здвигаються вправо на 1, 2 та 3 байти

відповідно.

$$\begin{pmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{pmatrix} \rightarrow \begin{pmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,1} & s_{1,2} & s_{1,3} & s_{1,0} \\ s_{2,2} & s_{2,3} & s_{2,0} & s_{2,1} \\ s_{3,3} & s_{3,0} & s_{3,1} & s_{3,2} \end{pmatrix}$$

Це перестановка елементів матриці, в якій приймають участь тільки елементи рядків, тому перетворення оборотно.

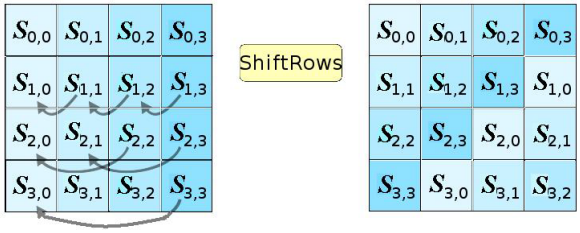

Рисунок 1.3 – Графічне представлення операції $ShiftRows$.

3. Операція $MixColumns$.

За допомогою цієї операції виконується перемішування байтів в стовпчиках матриці $State$. Кожен стовпець цієї матриці приймається як многочлен над полем $GF(2^8)$ і множиться на фіксований многочлен

$$c(x) = c_3x^3 + c_2x^2 + c_1x + c_0 = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$$

за модулем многочлена $x^4 + 1$ (всі коефіцієнти многочленів над полем $GF(2^8)$ — байти). Таку операцію можна записати в матричному вигляді:

$$\begin{pmatrix} c_0 & c_3 & c_2 & c_1 \\ c_1 & c_0 & c_3 & c_2 \\ c_2 & c_1 & c_0 & c_3 \\ c_3 & c_2 & c_1 & c_0 \end{pmatrix} \cdot \begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \cdot \begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix} = \begin{pmatrix} s'_0 \\ s'_1 \\ s'_2 \\ s'_3 \end{pmatrix}$$

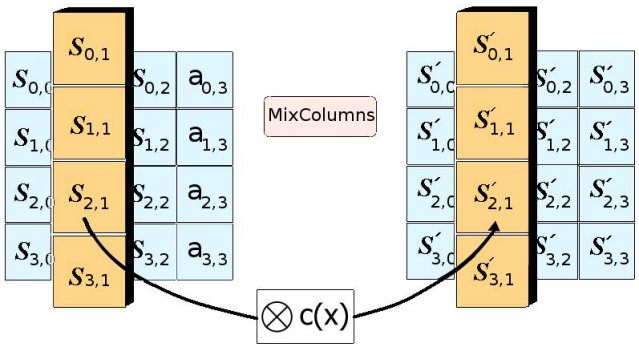

Рисунок 1.4 – Графічне представлення операції $MixColumns$.

Многочлен $c(x)$ - взаємнопростий з x^4+1 над полем $GF(2^8)$. Тому в полі існує обернений многочлен $c^{-1}(x)(\mathrm{mod} x^4+1) \Rightarrow$ матриця в цій формулі оборотна.

3. Операція $AddRoundKey$.

Функція $AddRoundKey(State,\ RoundKey)$ побітово складає елементи змінної $RoundKey$ і елементи $State$ по такому принципу: i-й стовпчик данних (i=0,1,2,3) складається з деяким 4-байтовим фрагментом ключа $W[4r+1]$, де r — номер поточного раунду алгоритма. При шифруванні перше складання ключа раунду відбувається до першого виконання операції $SubBytes$.

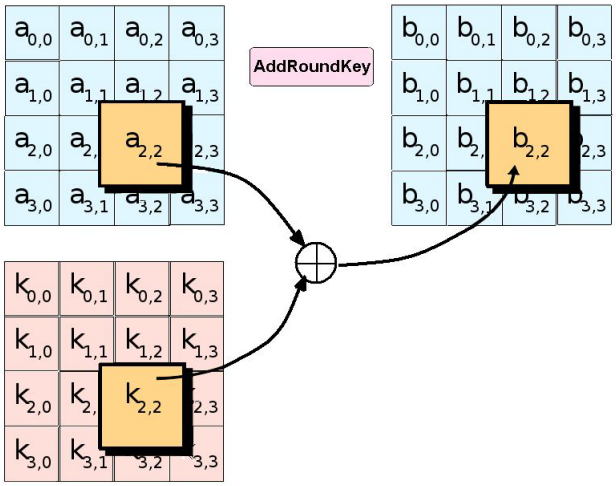

Рисунок 1.5 – Графічне представлення операції $AddRoundKey$.

Квантове використання диференційної властивості S-блоку AES

Далі представлено ефективний (по часу та пам'яті) спосіб вирішення диференційного рівняння S-блоку AES, котрий було описано в [10]. Зазвичай цією операцією нехтують, оскільки її можна вирішити за допомогою справочної таблиці 28×28. Однак, щоб зробити таку таблицю квантово доступною, потрібно кілька кілобайт qRAM, компоненту, котрий є дуже дорогим, тому автори вирішили на це не покладатися. Цей аналіз має важливе значення для атаки, котра була проведена в [10].

Інші підходи. Також було розглянуто різні способи вирішення данної проблеми. Можна послідовно протестувати всі пари (Δ_x,Δ_y). Всього їх 215, тому можна оцінити, що це буде в сто разів дорожче.

Подальші додатки. Цей аналіз може використовуватись для будь-якої атаки на AES, котра заснована на диференційному рівнянні S-блоку. Більше того, його можна узагальнити до інших S-блоків на основі оберненого.

1.5 Квантові атаки на AES

Квантове відновлення ключа — це процедура, котра відновлює ключ шивидше, ніж повний пошук, і не використовує всі можливості для ключа. Це визначення прямо слідує з класичного, хоча вого потребує більш глибокого занурення в деталі квантових обчислень та моделей противника.

Квантові схеми. Така схема записується як послідовність вентилів, котрі застосовуються до набору кубітів. Кубіти — двомірні квантові системи, записані в вигляді лінійної комбінації $|0\rangle$ і $|1\rangle$. Вони описуються вектором у двомірному просторі Гільберта H. Це означає, що для данної квантової схеми A, котра на вході $|0\rangle^n$ повертає деяку суперпозицію, можна подати обернену A в якості оператора A', і отримати $|0\rangle^n$. Ця операція вважається невираховною. Доволі часто

трапляється так, що підрахунок A повертає якийсь значимий результат, наприклад, біт, котрий необхідно зберегти, але використовує додаткові реєстри. Потрібно повернути стани цих реєстрів в $|0\rangle$, щоб повторно використати їх і обмежити загальне число кубітів(в даному випадку ці непостійні реєстри називаються допоміжними кубітами). Щоб зробити це просто копіюється вихідний результат A в вихідний реєстр і A не рахується для повторної ініціалізації допоміжних об'єктів.

Схематична модель стала стандартом в постквантовій криптографії, оскільки вона забезпечує загальну основу для порівняння квантових противників, незалежно від їх практичної реалізації. Складність часу, котра предтасвляє основний інтерес — це число вентилів схеми. Складність квантової па'яті — кількість кубітів. Це означає, що замість підрахунку класичних операцій і класичної пам'яті рахуються квантові вентилі і, можливо, максимально зменшується кількість використовуваних кубітів.

При розгляді і порівнянні квантових схем, необхідно опиратись на невеликий набір універсальних вентилів. Вентиль цього набору рахується однією операцією. Оскільки автори опирались на [11] для підрахунку квантових вентилів при вичерпному пошуці, використовується той же звичайний набір вентилів "Кліффорд+T". Група Кліффорда формується однокубітними вентилями Адамара:

$$H : H|b\rangle = \frac{1}{\sqrt{2}}|0\rangle + (-1)^b \frac{1}{\sqrt{2}}|1\rangle$$

однокубітне перетворення $S|0\rangle = |0\rangle$, $S|1\rangle = i|1\rangle$ і двокубітними вентилями CNOT: $CNOT|x\rangle|b\rangle = |x\rangle|x \oplus b\rangle$. T-вентиль також є однокубітним і додає фазу $e^{i\pi/4}$ на стан $|1\rangle : T|0\rangle = |0\rangle$ і $T|1\rangle = e^{i\pi/4}|1\rangle$. Використовуються вентилі Тоффолі, щоб не помилитися з T-вентилями. Він оборотно реалізує одну нелінійну операцію: $\text{Toffoli}|a\rangle|b\rangle|c\rangle = |a\rangle|b\rangle|c \oplus (a \wedge b)\rangle$. Вентилі ТОффолі можуть бути реалізовані з використанням 7 T-вентилів і 8 вентилів Кліффорда.

Далі буде описано результати, отримані в [10] після проведення найбільш відомих квантових атак на AES.

Квантово неможливі дифиренційні атаки.

Неможливі диференційні атаки використовують той факт, що деякі події можуть відбуватись в шифрі (наприклад, диференційний перехід, що передбачає неможливість). Це забезпечує відмінну ознаку для декількох середніх раундів, яка розширюється на кілька раундів назад і вперед, включаючи деякі ключові біти на шляху. Для хороших ключових здогадок подія відбуватиметься не за визначенням, а для поганих здогадок вона може відбутися. Отже, зловмисник просіює простір ключів, видаляючи неправильні здогадки про ключі, і правильним буде лише один — той, що залишиться.

Найкраща неможлива диференційна атака на AES-128 націлена на 7 раундів і забезпечує порівняний компроміс з деякими перевагами на кращі атаки meet-in-the-middle в [12].

Найефективніший спосіб побудови неможливих диференційних атак — спочатку отримати набір пар, що може призвести до неможливого середнього диференціалу, а далі — відкинути можливі ключі, пов'язані з кожною парою достатньо ефективним способом, завдяки методу раннього переривання. Хороший ключ буде серед тих, що не були відкинуті.

Було здійснено кілька спроб квантувати ці атаки. На сьогоднішній день жоден не виявився кращим ніж вже існуючі.

Виявляється, складно оптимізувати підрахунок пар за допомогою квантових обчислень: інтенсивно використовується класична пам'ять, зберігаючи структуру цілою, щоб ефективно отримати пари, котрі стикаються з очікуваними байтами на виході. Хоча квантовий пошук випакових функцій, як відомо, швидше чим класичний з використанням qRAM [13].

Використовуючи цей важкий для пам'яті метод, можна зменшити складність даних в моделі $Q2$ і прискорити обчислення пар.

Фаза просіювання може бути прискорена: при ключовій здогадці можна виконати квантовий пошук пар, які задовольняють неможливий диференціал. Це можна використати як тест на існування такої пари для

зовнішнього пошуку. Інші класічні покращення (такі як методи перевірки стану або множинні диференціали, як в [14]) потребували б конкретних квантових реалізацій.

Квантові Square атаки.

Square атака була запропонована в [27], і вивчена в оригінальному специфікаційному документі AES [9] орієнтованому на 6 раундів. Він був розширений до 7 раундів для AES-192 і 256 [16]. Він використовує інтегральне розпізнавання на 3 раунді AES, що потребує 256 обраних відкритих текстів (якщо байт приймає всі можливі 28 значень, а інші залишаються постійними, через три раунди всі байти внутрішнього стану будуть в балансі). Він розширюється додаванням декількох раундів до і після нього, ціною підвищеної складності даних (232 обраних відкритих текстів) та деякими здогадками про байти ключів. Це сімейство атак гірше ніж самі відомі атаки типу meet-in-the-middle [17], але вони цікаві, оскільки забезпечують низьку складність. Атаки з низьким рівнем данних (наприклад в порівнянні з атаками DS-MITM) представляють незалежний інтерес, про що свідчать нові тенденції, такі як [18].

Ця атака являється квантовою атакою для 6-раундового AES-128 в тому сенсі, що вона коштує менше часу, чим вичерпний пошук Гровера (приблизно 264 шифрування).

Були запропоновані квантові версії square-атаки. Вони виконуються в моделі $Q1$, в котрій важливо використовувати метод часткових сумм з [16]. Не було знайдено способу зробити так, не покладаючись на qRAM. Це основне обмеження результатів, наведених в таблиці 1.4.

Таблиця 1.4 – Оцінки необхідних ресурсів для деяких атак.

Версія	Класичний аналог	Запити	Кв. час	Кв. пам'ять	Класична пам'ять	Алг. Гровера по ключам
6-р. AES-128	[FKL$^+$00]	2^{35}	2^{44}	2^{25}	2^{36}	$2^{72.2}$
7-р. AES-256	[DKR97]	2^{37}	2^{121}	незначна	2^{38}	$2^{137.3}$
7-р. AES-256	[FKL$^+$00]	2^{37}	2^{107}	2^{27}	2^{38}	$2^{137.3}$
7-р. AES-192	[FKL$^+$00]	2^{37}	$2^{103.4}$	2^{27}	2^{38}	$2^{105.6}$

Квантова DS-MITM атака.

DS-Meet-in-the-Middle атака була введена в [19] для аналізу AES. З тих пір було запропоновано багато покращень. Найбільш ефективне з них, на AES зі зменшеною кількістю раундів, описано в [17].

В цій атаці також використовується розпізнавач в середніх раундах. В цьому випадку розпізнавач розглядає (малий) набір можливих вхідних даних для середніх раундів, так що, якщо один з набору вхідних даних слідує певному диференційному маршруту, набір можливих пов'язаних значень для частини стану на виході буде мати обмежену кількість можливостей (набагато меншу, ніж в випадковому випадку). Ця відмінна ознака також може бути розширена на кілька циклів назад і вперед, включаючи деякі біти секретного ключа. Атаки запропоновані раніше завжди будуються наступним чином: спочатку, всі можливі набори входів-виходів для розпізнавання середніх раундів обчислюються і зберігаються. Далі, в онлайн фазі подається запит на пари входів. Кандидати, котрі слідують по диференційному маршруту, зберігаються, і для кожного виконується вичерпний пошук по задіяним бітам ключа, обчислюючи відповідний середній набір. Далі, перевіряється, чи зберігаються ці значення в заздалегіть вирахуваній таблиці. Якщо це так, то знайдено кандидата на секретні біти ключа.

Опис алгоритму Гровера.

Розглянемо алгоритм Гровера, описаний в [20]. Він приймає в якості входу квантову схему, котра реалізує булеву функцію: $f : \{0,1\}^k \to \{0,1\}$ звичайним методом, тобто, через квантову схему U_f, котра реалізує $|x\rangle|y\rangle \mapsto |x\rangle|y \oplus f(x)\rangle$, поки $x \in\{0,1\}^n$ і $y \in\{0,1\}$. Базовий алгоритм Гровера знаходить такий x_0, для якого $f(x_0) = 1$. Позначимо Адамарову трансформацію 2×2 як H, алгоритм Гровера базується на багаторазовому повторенні G до початкового стану $|\psi\rangle \otimes |\varphi\rangle$, де $|\psi\rangle = \frac{1}{\sqrt{2^k}} \sum_{x\in\{0,1\}^k} |x\rangle$, $|\varphi\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$, де

$$G = U_f((H^{\otimes k}(2|0\rangle\langle 0| - 1_{2^k})H^{\otimes k}) \otimes 1_2), \text{ де}$$

$|0\rangle$ позначає всі нульові базисні стани відповідного розміру. Загалом, G має бути застосовано $O(\sqrt{N/M})$ разів для визначення такого x_0, що $f(x_0) = 1$ з постійною ймовірністю, де N - загальне число кандидатів, тобто $N = 2^k$, і при умові, що є саме M рішень, тобто $M = |\{x : f(x) = 1\}|$. Якщо ми знаємо одне рішення $M = 1$, це означає, що ми можемо вирішити, застосовуючи $H^{\otimes k+1}$ до початкового стану $|0\rangle^k \otimes |1\rangle$, далі G^ϱ, де $\varrho = \lfloor \frac{4}{\pi}\sqrt{N} \rfloor$, з майбутнім виміром всього квантового регістру, котрий дасть рішення x_0 з високою ймовірністю [21],[22].

Як було показано в [10], дійсно можна визначити функцію f з набору можливих ключів, тобто k ={128,192,256} для випадку AES, тому що точно є одне вирішення проблеми пошуку правильного ключа k, який використовується для розшифрування невиликого набору заданих пар відкритого та зашифрованого текстів, тобто ми можемо (правдоподібно) реалізувати ситуацію $M = 1$, визначивши функцію f, котра нас влаштує. Алгорим Гровера можна модифікувати різними способами, щоб він міг впоратися з великим (але відомим) числом вирішень: якщо число M відомо, $O(\sqrt{N/M})$ ітерацій достатньо, однак, якщо це число невідомо, виникає проблема, що неможливо вибрати правильну кількість ітерацій. Тим не менш, існує варіант алгоритму, який знаходить вирішення в очікуваному часі виконання $O(\sqrt{N/M})$, навіть коли число рішень M невідомо[21].

Далі було розглянуто кількість вентилів і вимоги до простору, що необхідні для реалізації алгоритму. Розглянемо схему з частини (b) Рисунку 1.6, і проаналізуємо її складність. Хоча і визначається операцією Кліффорда, окрім U_f котра включає в себе класичні розрахунки функцій AES, також необхідно було визначити вартість операції $(2|0\rangle\langle 0| - 1)$. Це зводиться до реалізації k-кратно контрольованого вентиля NOT, де для нас k $\in$128,192,256. Оцінки для цих вентилів в термінах вентилів Тоффолі було отримано в [23], (для $n \geqslant 5)8k - 24$ вентилів Тоффолі, котрі оцінюються в 1.000, 1.512, і 2.024 вентилі Тоффолі для фази операції $(2|0\rangle\langle 0| - 1_{2^k})$ відповідно. Для числа

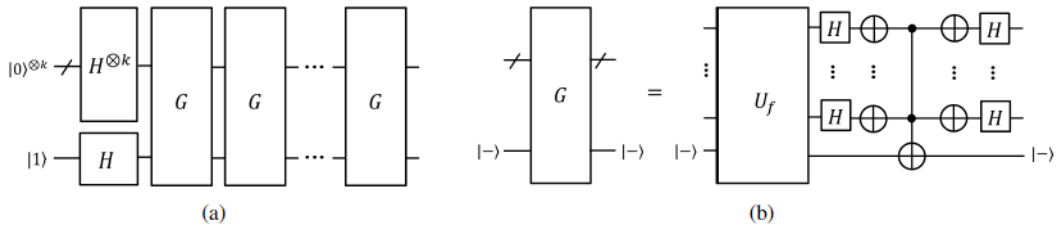

Рисунок 1.6 – (a) Квантова схема реалізації алгоритму Гровера. Алгоритм складається зі створення суперпозиції $\sum_x |x\rangle$ в верхньому реєстрі, котрий для випадку AES має $k = 128; 192; 256$ кубітів і единий стан — кубіт $|-\rangle = |0\rangle - |1\rangle$ в нижньому регістрі. Оператор G являється ітерацією Гровера і використовується кількість разів, залежно від $\lfloor \frac{\pi}{4}\sqrt{2^k} \rfloor$ (b) Один раунд алгориму Гровера. Показаний оператор $G = U_f((H^{\otimes k}(2|0\rangle\langle 0| - 1_{2^k})H^{\otimes k}) \otimes 1_2)$ і схема його розкладу. Ефект вентилів мід двома рівнями вентилів Адамара базується на інверсії фази базового стану $|0\rangle$ на старших k бітах.

вентилів T і Кліффорда можна застосувати верхню межу, помноживши k на 7, також можна показати верхню границю $32k - 84$ для k-кратно контрольованих вентилів NOT, тобто було отримано 4.012, 6.060, і 8.108 для T за операцію фази для трьох розмірів ключа k ∈128,192,256.

Також було отримано оцінку для $f : \{0,1\}^k \to \{0,1\}$ котра продовжується першим відображенням $K \mapsto (AES_k(m_1),...,AES_k(m_r))$ і далі обчислення рівності функції результуючого вектора з заданим шифротекстом $c_1,...,c_r$, де $c_i \in \{0,1\}^{128}$. Іншими словами, знаходиться значення функції f для заданого вхідного ключа $K \in \{0,1\}^k$(де k ∈128,192,256):

$$f(K) := (AES_K(m_1) = c_1) \wedge ... \wedge (AES_K(m_r) = c_r).$$

Правдоподібно, що r =3;4;5 достатньо для трьох стандартних розмірів ключів AES. Функція рівності може бути реалізована за допомогою вентилю NOT, котрий має $128r$ та одну ціль. Використання наведених формул приводить до того, що кількість вентилів Тоффолі 3.048, 4.072, і

5.096 відповідно, а кількість T-вентилів - 12.204, 16.300 і 20.396 відповідно.

Реалізація логічного предикату - тестування ключа.

Важливим компонентом, необхідним в алгоритмі є схема, котра при вводі кандидату $|K\rangle$ показує, чи рівний цей ключ секретному цільовому ключу. Ідея в тому, щоб зашифрувати якийсь фіксований відкритий текст і порівняти результат з відповідним зашифрованим текстом (який вважається відомим) з цільовим секретним ключем.

Оскільки AES завжди працює з 128-бітними відкритими текстами, принаймні для 192- і 256-бітних ключів ми повинні припустити, що фіксації однієї пари відкритого та зашифрованого текстів недостатньо для точного визначення секретного ключа. Сперичаючись з суворим лавинним ефектом, можна правдоподібно припустити, що для кожної пари ключів $(K,K') \in \{0,1\}^{k\times k}$, $K \neq K'$ умова:

$$(AES_K(m_1),...,AES_K(m_r)) \neq AES_{K'}(m_1),...,AES_{K'}(m_r)$$

справедлива для деякого набору відкритого тексту $m_1,...,m_r$. Причина цього заключається в тому, що для фіксованого відкритого тексту, при заміні біта в секретному ключі, кожен біт відповідного шифротексту має змінюватися з вірогідністю $1/2$. Відповідно, для r одночасних пар відкритого і зашифрованого текстів, котрі зашифровані двома секретними ключами $K \neq K'$, ми очікуємо отримати різні результати, з ймовірністю $1 - 2^{-rn}$ (де n — довжина повідомлення), якщо відкриті тексти попарно відрізняються. Відповідно, з $2^{2k} - 2^k$ пар ключів (K,K') (де $K \neq K'$), очікується, що $(2^{2k} - 2^k)2^{-rn} \leq 2^{2k-rn}$ ключів дадуть ті самі розшифрування. Здається правдивою оцінка того, що

$$r > \lceil 2k/n \rceil$$

відкритих текстів буде достатньо, щоб гарантувати, що для кожного $K' \neq K$ буде доступний хочаб один відкритий текст. Оскільки AES має 128-бітні відкриті тексти, ми маємо $n = 128$, тобто попередня оцінка означає, що для довжини ключа k у противника є $r > \lceil 2k/128 \rceil$ пар з

відкритого, та зашифрованого тексту $(m_1,c_1),...,(m_r,c_r)$ для доступного цільового ключа. Іншими словами, щоб унікально охарактеризувати секретний ключ, ми вважаємо, що $r = 3(AES-128)$, $r = 4(AES-192)$ і $r = 5(AES-256)$ відповідних пар відомо супротивнику.

Висновки до розділу 1

В даному розділі було описано алгоритми AES та Калина та розглянуто деякі атаки на ці шифри. Видно, що ці два шифри дуже схожі між собою, що і буде використано при дослідженні.

2 ОГЛЯД ОЦІНОК РЕСУРСІВ, НЕОБХІДНИХ НА РЕАЛІЗАЦІЮ КРИПТОГРАФІЧНИХ АЛГОРИТМІВ В КВАНТОВІЙ МОДЕЛІ ОБЧИСЛЕНЬ

В даному розділі проаналізовані існуючі наукові джерела, в яких було проведено дослідження оцінок необхідної кількості квантових ресурсів необхідних на реалізацію деяких криптографічних примітивів в квантовій моделі обчислень.

2.1 Схеми для основних операцій шифру AES

Внутрішній стан AES складається з 128 біт, котрі складені в масив 4×4 байтів. Потрібно виділити 128 кубітів для зберігання поточного внутрішнього стану.

AddRoundKey. В реалізації цієї операції ключа було гарантовано, що поточний раундовий ключ буде доступний на 128. Реалізація побітового XORу ключа зменшує до 128 кількість вентилів $CNOT$, поки всі можуть виконуватися паралельно.

MixColumns. Оскільки ця операція працює з цілими стовпцями станів (або 32 бітами) за раз, матриця, яка була вказана NISTом в [24], використовувалась для генерації матриці 32×32. Розклад LUP було використано для цієї матриці, щоб обчислити цю операцію з 277 вентилями CNOT і загальною глибиною, рівною 39. Було запропоновано аналогічну, але меншу версію розкладу типу LUP, котра була використана.

ShiftRows. Ця операція складає перестановку поточного стану AES, не потрібно додавати ніякі елементи для реалізації цієї операції, оскільки вона відповідає перестановці кубітів. Замість цього було просто скоректовано положення наступних вентилів, для того, щоб переконатися,

що використовується правильний вхідний дріт.

SubBytes. Ця операція виконує заміну одного байт поточного стану новим значенням. Для класичної реалізації справочна таблиця може бути гарним варіантом, але в даному випадку явний підрахунок результату буде використовувати менше ресурсів. Обробляючи байт стану, як елемент $\alpha \in F_2[\mathrm{x}]/(1 + x + x^3 + x^4 + x^8)$, спочатку потрібно знайти мультиплікативну обернену α. Після слідує афінне перетворення, далі обчислюється

$$\alpha^{-1} = \alpha^{254} = ((\alpha \cdot \alpha^2) \cdot (\alpha \cdot \alpha^2)^4 \cdot (\alpha \cdot \alpha^2)^{16} \cdot \alpha^{64})^2$$

Знову використовуючи розклад LUP, відповідне множення матриць може бути виконане на місці з використанням лише вентилів CNOT. І, відповідно, регулюючи положення наступних вентилів, розуміючи, що перестановка безкоштовна, не потрібно вводити нові вентилі.

З прикладу 2.1 видно, що піднесення в квадрат може бути реалізоване тільки з 12 вентилями CNOT. Результуюча схема показана на Рисунку 2.1. Щоб реалізувати шість множень в формулі **??**, використовується множник загального призначення в двійковому полі. Було обрано дизайн Маслова з [25], котрий потребує на 60% кубітів менше, ніж в [26]. Це відбувається за рахунок підвищеної складності вентилів, і можна було б розглянути інший вибір конструкції. Для конкретного представлення поліноміального базису F_{256} дизайн Маслова потребує 64 Тоффолі і 21 CNOT вентилів, котрі можна перетворити в $64 \cdot 7 = 448$ T-вентилів та $64 \cdot 8 + 21 = 533$ вентилів Кліффорда.

Приклад 2.1.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot$$

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

В формулі три множення є дублікатами, тому, для реалізації обернення достатньо чотири множення. Намагаючись зменшити кількість кубітів, потрібних на кожному кроці, фактичний розрахунок α^{-1} займає 40 кубітів, даючи на виході $|\alpha\rangle$, $|\alpha\rangle^{-1}$ і 24 реініціалізованих кубіти. Для цього і реініціалізації кубітів інвестується дванадцять лінійних

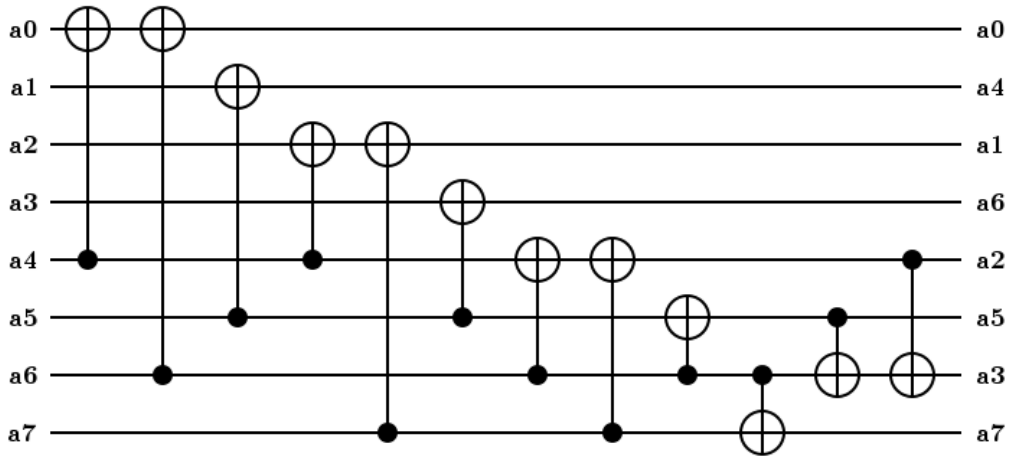

Рисунок 2.1 – Піднесення до квадрату в $F_2[\mathrm{x}]/(1 + x + x^3 + x^4 + x^8)$.

перетворень і вісім F_{256}-множень, загалом 3584 T-вентілі і 4539 вентилів Кліффорда.

Як тільки α^{-1} знайдено, має бути обчислено афінне перетворення, це можна зробити за допомогою розкладу типу LUP; чотири вентилі NOT відповідають за складання вектора після множення на матрицю. Всього для одного 8-бітного S-блоку потрібно 3584 T-вентилі та 4569 вентилів Кліффорда.

Альтернативна реалізація *SubBytes* для мінімізації кількості кубітів. Обернення $\alpha \mapsto \alpha^{-1}$ (де відображається 0 в 0) можна розглядати як перестановку на F^{256}. Ця перестановка непарна, в той час, як квантові схеми з вентилями NOT, CNOT і Тоффолі на $n > 3$ кубітів генерують повну групу A_{2n} парних перестановок. Відповідно, потрібно використати один допоміжний кубіт, тобто, всього їх буде використано 9. Отже, було знайдено схему, в котрій не більше 9695 T-вентилів і 12631 вентиля Кліффорда, що майже в три рази більше за те, що було пораховано раніше, але з використанням всього 9 кубітів, замість 40.

Розширення ключа.

Стандартна реалізація розширення ключа для AES (k = 128,192,256) розділяє k-бітний ключ на 4, 6 або 8 слів довжиною 32, і повинна розширювати k-бітний ключ на 44 слова для k=128, 52 слова для

k=192,та 60 слів для k=256. Кожне розширення ключа AES використовує одні і ті ж операції, і в реальній конструкції раундових ключів є тільки невеликі відмінності. Це такі операції як $RotWord$, $SubBytes$, і $Rcon[i]$, котра додає $x^{i-1} \in F_{256}$ до першого байту кожного слова.

В той час, коли три різні версії AES використовують до 14 раундів розрахунків, розширення ключа не залежить від входу. Слова, створенні розширенням ключа, були розділені на дві категорії: слова, яким потрібні $SubBytes$ для своїх підрахунків, і слова, котрим це не потрібно. Слова, котрі не включають в себе $SubBytes$, можуть бути рекурсивно побудовані з тих, котрі створюються за допомогою операції XOR, заощаджуючи за рахунок цього до 75% вартості зберігання розширеня ключа. Найдорожчим є слово w_{41} в AES-128, котре побудовано за допомогою ксору одинадцяти попередніх слів, його вартість складає 352 CNOT вентилі та глибину 11.

Таблиця 2.1 – Оцінки квантових ресурсів для фази розширення ключів AES.

	К-сть вентилів			Глибина		К-сть кубітів	
	NOT	CNOT	Тоффолі	T	Загальна	Зберігання	Допоміжні
128	176	21.448	20.480	5.760	12.636	320	96
192	136	17.568	16.384	4.608	10.107	256	96
256	215	27.492	26.624	7.488	16.408	416	96

Оскільки $SubBytes$ коштує дорого, інші слова зберігаються у міру їх побудови. У класичній реалізації AES ці слова (кожне четверте чи шосте) утворюються, починаючи з попереднього слова, однак у цій конструкції попереднє слово має бути побудовано та видалено за потребою. Наприклад, в AES-128 для побудови w_8 спочатку потрібно побудувати w_7 так: $w_7 = w_4 \oplus w_3 \oplus w_2 \oplus w_1$.

Це може бути зроблено на раніше побудованому слові (в цьому випадку w_4), заощаджуючи кубіти, вентилі та глибину. Оскільки побудова w_8 потребує використання w_4, наведений вище процес потрібно повторити, до кінця побудови w_8. Для побудови цих слів, аналогічно $ShiftRows$,

RotWord може бути виключено, якщо положення вентилів зміщене. Так як SubWord незалежно застосовує *SubBytes* до кожного байту слова, кожне з чотирьох обчислень *SubBytes* може виконуватися одночасно.

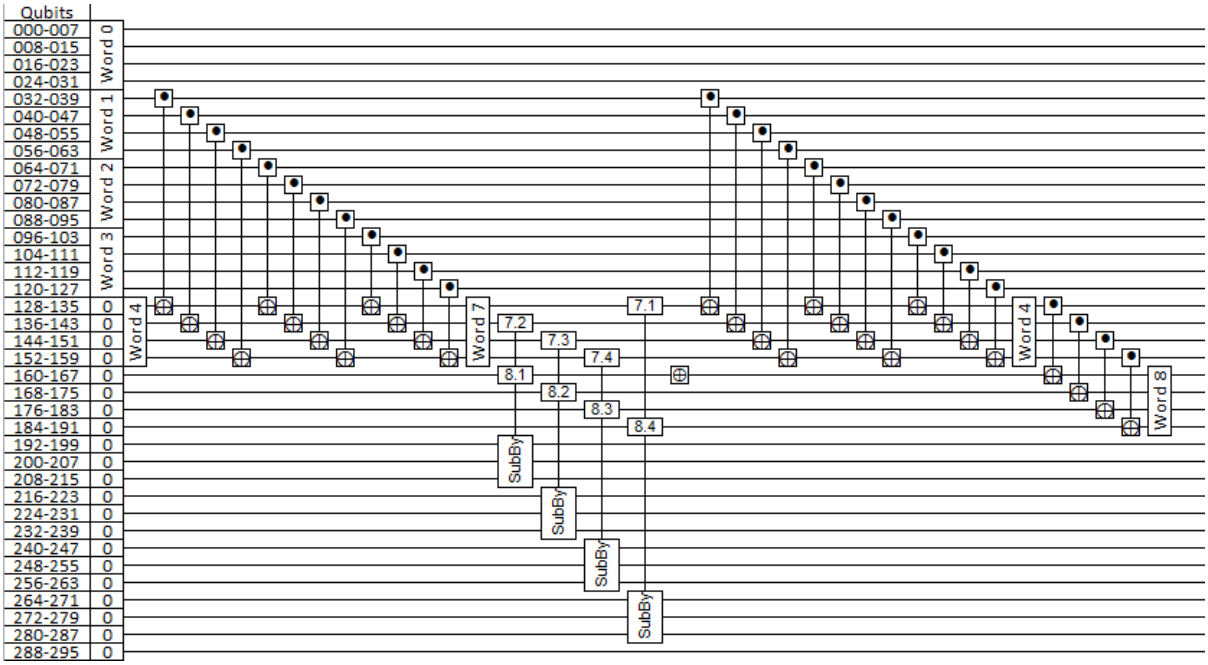

Рисунок 2.2 – Кострукція w_8.

Щоб дозволити кожному з чотирьох *SubBytes* шаблонів за раунд виконуватись одночасно, потрібно 96 допоміжних кубітів та 32 на зберігання нового слова. З кожним побудованим словом, потребуючим попереднє слово буде побудовано першим, далі глибина залишилась незмінна. Витрати на обчислення наведені в Таблиці 2.1 (перечислені вистрати кубіт не включають зберігання початкового ключа).

Раунди AES.

AES починається з простого кроку - XORy вхідних з першими чотирьма словами ключа. Оскільки в цьому випадку вхід являється фіксованим значенням, і додавання фіксованого значення може бути виконано простим переключенням бітів, приблизно 64 вентилі NOT використовуються в перших чотирьох словах ключа для початку першого раунда. При необхідності це може бути відмінено пізніше. Якщо це не так, то для зберігання вхідних даних потрібно 128 кубітів, а для

обчислення цього кроку можна використати 128 вентилів CNOT. Хоча всі 10, 12 або 14 раундів AES використовують одні й ті самі базові функції, структура схеми трохи відрізняється в кожному раунді для зменшення кількості необхідних кубітів та глибини. $SubBytes$ повинні рахуватись 16 разів за раунд, для цього необхідно 384 допоміжні кубіти для всих, виконуваних одночасно, або ж необхідне збільшення глибини. Використовуючи тільки 24 допоміжні кубіти і 128 кубітів, необхідних для зберігання результату, було помічено, що всі 16 обчислень $SubBytes$ за раунд може бути виконано з максимальною глибиною 8.

Оскільки $SubBytes$ не виконуються на місці, і AES-k потребує 128 кубіт на раунд, підрахунок займає 128 кубітів, помножених на кількість раундів AES, на додачу до кількості кубітів, необхідних для зберігання початкового ключа. Це число може бути зменшено шляхом зміни кроків між підрахунками, щоб очистити кубіти для подальшого використання. Після використання $SubBytes$, вхід може бути видалено шляхом зміни достатньої кількості кроків (але вихід не може бути видалений, оскільки його аналог (обернений) зник). Оскільки AES-128 використовує 10 раундів, використовуючи 512 кубітів для зберігання і 24 допоміжні кубіти, дозволяє примінити оборотний процес три рази. Для AES-192 і AES-256 було використано 640 кубітів для зберігання, оскільки не вдалось провести три раунди обернення на 536 кубітах.

Для AES-192 і AES-256 процес обернення виконується після п'ятого, дев'ятого та дванадцятого раундів, (рис. 2.3) потребуючи тільки на 128 кубітів більше, ніж AES-128.

Як вже вказувалось раніше, $ShiftRows$ безкоштовний, і викоростання декомпозицій типу LUP для $MixedColumns$ дозволяє виконати цей процес з використання 277 вентилів CNOT і глибиною 39. Дли обчислення всіх десяти раундів AES-128 потрібно було 536 кубітів, 664 кубіти для дванадцяти раундів AES-192 і чотирнадцяти раундів AES-256.

XOR раундових ключів може бути виконано безпосередньо над

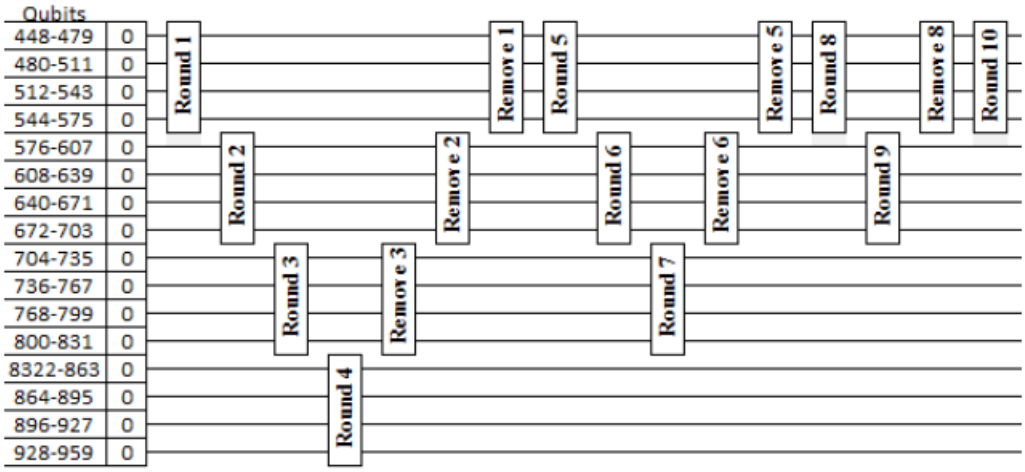

Рисунок 2.3 – Представлення процесу обернення для AES-128. Цей метод залишає 4, 7 і 9 раунди без можливості видалення, якщо весь процес не буде повністю змінено.

входом для кожного раунда. Якщо необхідний ключ раунду вже створено, для завершення раунду використовується 128 вентилів CNOT з глибиною 1. Якщо раундовий ключ ще не створено, і, відповідно, він являється комбінацією вже створених ключів, то необхідно, щоб цей процес виконувався тільки кілька разів. AES-128 потребує, щоб це було зроблено 11 разів (найбільше) в випадку w_{41}, збільшуючи глибину і кількість вентилів CNOT максимум на 11.

Оцінка ресурсів для оборотної реалізації AES.

Далі буде наведено три Таблиці(2.2, 2.3 та 2.4), в яких показано оцінку необхідних вентилів, глибини та кубітів, яка була отримана Грасселом та іншими[27].

Таблиця 2.2 – Витрати квантових ресурсів на реалізацію AES-128.

	К-сть вентилів		Глибина		К-сть кубітів
	T	Кліффорда	T	Загальна	
Поточний стан	0	0	0	0	128
Генерація ключів	143.360	185.464	5.760	12,626	320
10 раундів	917.504	1.194.956	44.928	98.173	536
Загальна	1,060,864	1.380.420	450.688	110.799	984

Таблиця 2.3 – Витрати квантових ресурсів на реалізацію AES-192.

	К-сть вентилів		Глибина		К-сть кубітів
	T	Кліффорда	T	Загальна	
Поточний стан	0	0	0	0	192
Генерація ключів	114.688	148.776	4.608	10.107	256
12 раундів	1.089.536	1.418.520	39.744	86.849	664
Загальна	1.204.224	1.567.296	44.352	96.956	1.112

Таблиця 2.4 – Витрати квантових ресурсів на реалізацію AES-256.

	К-сть вентилів		Глибина		К-сть кубітів
	T	Кліффорда	T	Загальна	
Поточний стан	0	0	0	0	256
Генерація ключів	186.368	240.699	7.488	16.408	416
14 раундів	1.318.912	1.715.400	52.416	114.521	664
Загальна	1,505.280	1.956.099	59.904	130.929	1.336

Для AES-192 (Таблиця 2.3) потріно менше кубітів і менша глибина ніж для AES-128 (Таблиця 2.2), тому що використовується додатковий простір для зберігання проміжних результатів і розпаралелення частин ланцюга.

Оцінка квантових ресурсів для AES зі зменшеною кількістю раундів

Перш за все, необхідно зосередитися на оцінках квантових ресурсів AES. Кількість квантових вентилів для компонентів AES та версій зі зменшеним циклом було отримано з [11]. Дійсно, такі точні підрахунки необхідні для оцінки того, чи являється процедура відновлення ключа, за заданий час, атакою чи ні.

Точні квантові оцінки ресурсу для AES були зроблені в [11], з різними технічними прийомами по зменшенню кількості необхідних кубітів. Оборотна реалізація S-блоку AES в [11] коштує 3584 T-вентилі, 4569 вентилів Кліффорда і 40 кубітів. Оскільки це єдиний нелінійний компонент AES, він також є найбільш витратним. Наприклад операція $ShiftRows$ відповідає лише перенумерації кубітів, тому фактично не потребує ресурсів. Крок $MixColumns$ також. Це відноситься як до вичерпного пошуку, так і до деяких атак, оскільки вони використовують

оборотні компоненти AES [11] практично в однакових кількостях.

Щоб звести до мінімуму кількість використовуваних допоміжних кубітів (кількість яких могла бути в 128 разів більше кількості раундів), автори [11] рахують раунди всередині чорного ящику AES. Тести в еквівалентних (оборотних) S-блоках наведені в таблиці 2.5.

Таблиця 2.5 – Кількість необхідних S-блоків.

Компонент	Кількість S-блоків	Кубіти
S-блок	1	40
128-бітний список ключів(10 раундів/10 ключів)	40	320
192-бітний список ключів(12 раундів/8 ключів)	32	256
256-бітний список ключів(14 раундів/7 ключів)	52	664
6-раундовий AES	144	408
7-раундовий AES	160	536
8-раундовий AES	192	536
6-раундовий AES(з списком ключів)	168	856
8-раундовий AES(з списком ключів)	224	1200

Оцінка використання квантових ресурсів алгоритмом Гровера

Було отримано оборотну схему для підрахунку $AES_K(m_i)$, тобто контур C, котрий реалізує $|K\rangle|0\rangle \mapsto |K\rangle AES_K(m_i)\rangle$. Загальна схема для реалізацаї U_f показана на рис. 2.4. Рівні AES можна застосовувати паралельно, однак, оскільки використані допоміжні кубіти повинні після кожного раунда повертатися чистими, ми повинні вираховувати кожен блок AES в кожному раунді. Відповідно, глибина (T) збільшується в два рази при кожному виклику U_f. Загальна кількість вентилів з іншої сторони збільшується в два рази, так як всі блоки тепер повинні бути обчислені. Кількість кубітів подається як r помножена на кількість кубітів всередині кожного блоку AES.

Після того, як блоки AES будуть обчислені, результат порівнюється

з заданим шифротекстами c_1,...,c_r. Слід звернути увагу на те, що AES працює з відкритими/шифро-текстам довжини 128, $c_i \in \{0,1\}^{128}$. Порівняння виконується за допомогою множення вентилю NOT, і елементи керування мають значення 0 або 1 в залежності від бітів c_i. Це позначається верхнім індексом c_i над елементами керування на рис.2.4. Тепер можна зібрати все до купи, щоб оцінити вартість алгоритму Гровера на основі оцінок ресурсів AES-k, наведених раніше: означення s_k загальним числом кубітів, t_k загальним числом T-вентилів, c_k загальною кількістю вентилів Кліффорда, δ_k — загальна T-глибина і Δ_k — загальна глибина, де k ={128,192,256}. Вимоги до простору складають $3s_{128}+1$ для AES-128, $4s_{192}+1$ для AES-192 і $5s_{256}+1$ для AES-256.

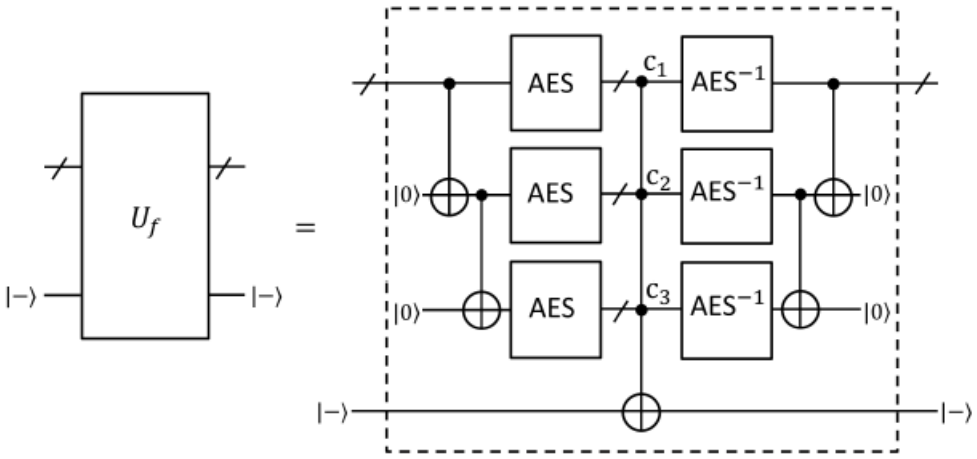

Рисунок 2.4 – Обернена реалізація функції U_f. Розглядається розмір ключа $k = 128$, для котрого достатньо $r = 3$ виклики AES, щоб зробити цільовий ключ унікальним. Для випадків $k = 192$ та $k = 256$ – $r = 4$ і $r = 5$ викликів відповідно. Загальна схема залишається спільною для трьох розмірів ключа.

Що стосується складності часу, було отримано, що для кожної ітерації алгоритму Гровера потрібно $6t_{128}$ T-вентилів для AES-128 плюс кількість T-вентилів, необхідну для 384-кратного контрольованого NOT

всередині U_f і 128-кратного контрольованого NOT для реалізації фази $(2|0\rangle\langle 0| - 1)$. В цілому, потрібно виконати $\lfloor \frac{\pi}{4} 2^{k/2} \rfloor$ ітерацій, тобто оцінка загальної кількості T-вентилів для алгоритму Гровера на AES-128 буде такою:

$$\lfloor \tfrac{\pi}{4} 2^{64} \rfloor \cdot (6t_{128} + 13.204) = 9.24 \cdot 10^{25} = 1.19 \cdot 2^{86}$$

Точно так само, можна оцініки кількість вентилів Кліффорда, котру, для простоти було прийнято рівною $6c_{128}$, пропускаючи деякі з вентилів, використаних під час раундів. Для AES-192 потрібно виконати $\lfloor \frac{\pi}{4} 2^{96} \rfloor$ ітерацій і $\lfloor \frac{\pi}{4} 2^{128} \rfloor$ ітерацій для AES-256, відповідно. В цілому, для алгоритму Гровера на AES-192 було отримано оцінку в $3.75 \cdot 10^{36} = 1.81 \cdot 2^{114}$ T-вентилів і для алгоритму Гровера на AES-256 $4.03 \cdot 10^{45} = 1.41 \cdot 2^{151}$ T-вентилів. Для загальної глибини схеми було отримано число раундів, помножене на $2\delta_k$, відповідно Δ_k, ігноруючи деякі елементи, котрі неістотно впливають. Загальні оцінки наведені в таблиці 2.6.

Таблиця 2.6 – Оцінки квантових ресурсів для алгоритму Гровера на AES-k

	К-сть вентилів		Глибина		К-сть кубітів
k	T	Кліффорда	T	Загальна	
128	$1.19 \cdot 2^{86}$	$1.55 \cdot 2^{86}$	$1.06 \cdot 2^{80}$	$1.16 \cdot 2^{81}$	2.953
192	$1.81 \cdot 2^{118}$	$1.17 \cdot 2^{119}$	$1.21 \cdot 2^{112}$	$1.33 \cdot 2^{113}$	4.449
256	$1.41 \cdot 2^{151}$	$1.83 \cdot 2^{151}$	$1.44 \cdot 2^{144}$	$1.57 \cdot 2^{145}$	6.681

2.2 Оптимізація кількості квантових ресурсів, необхідних для реалізації AES.

Однак в [28] була представлена нова схема для реалізації SubByte, яка базується на результаті Бояра і Перальта [29]. Такий підхід дозволяє значно зменшити кількість необхідних T-вентилів, в порівнянні зі схемами,

які запровонували Грассел[27] і Алмаяорі[30]. Їх схема потребує 32 кубіти, 55 вентилі Тоффолі, 314 вентилів CNOT, 4 вентилі NOT, глибину Тоффолі 40 і загальну глибину (NCT) 298, включаючи очистку допоміжних кубітів. В тому числі, на основі [27] та [30] було представлено нові квантові схеми для всих трьох довжин ключа, які одночасно забезпечують оптимізацію кількості кубітів, вентилів Тоффолі і Кліффорда.

Оптимізація для AES-128

В рамках розширення ключа розраховуються різні ключові слова k_i. Кожний k_4n потребує використання чотирьох S-блоків і XORy попередніх ключів. Після трьох раундів мають схожу структуру. Зберігаються k_{4n+3} після того, як раунд n буде повністю завершено. Щоб зберегти глибину, кожне ключове слово буде побудоване одночасно з раундом, в якому воно використовується, за винятком першого раунду. Це пов'язано з тим, що відкритий текст і ключ шифрування ксоряться для створення нульового раунда. Для побудови першого раунда необхідні обидва, тому перший раунд і k_n повинні створюватися послідовно. Для залишившихся раундів розпаралелювання значно зменшує глибину. Наприклад, під час другого раунда всі обчислення S-блоку для k_8, а також другий раунд можуть бути вирахувані з глибиною S-блоку рівній одиниці, з використанням допоміжних 320 кубітів. Як тільки ці S-блоки і MixColumns обчислені, k_8 може бути заксорений на перший раунд, за яким слідує побудова k_9, k_{10} і k_{11}, кожний з яких заксорений на першому раунді. Таким чином, другий раунд обчислюється повністю, а k_{11} зберігається, і вся ця конструкція займає глибину S-блоку, рівну одиниці. Коли 320 додаткових кубітів недоступні, не всі S-блоки можуть виконуватись паралельно, і глибина повинна бути збільшена до 7. Перший раунд (без k_4), другий раунд, видалення першого та п'ятого раунду розраховуються з глибиною S-блоку, рівній одиниці.

Хоча підрахунок ключових слів разом з раундами сильно зменшує загальну глибину, це означає, що коли допоміжні кубіти недоступні, для 20 S-блоків (16 для раунду і 4 для ключа) може знадобитися підрахунок

збільшеної глибини S-блоків. В той час, коли другий раунд має глибину S-блоку, рівну одиниці, сьомий раунд - 7, оскільки доступно всього 16 додаткових кубітів (плюс будь-які кубіти, які все ще не зберегли частину 7-го раунду). Іноді, раундові ключі можуть бути обчислені під час очистки попередніх раундів. Наприклад, при оберненні і очистці восьмого раундау може бути обчислений ключ для десятого (k_{40}), тому для обчислення десятого раунду потрібно всього 16 S-блоків з глибиною, рівною шести. І таким чином буде завершено обчислення AES-128.

Зберігаючи k_{4n+3} для кожного $n \leq 7$, при наближенні до 7-го раунда, і при збереженому k_{31}, можна видалити слова k_{15}, k_{11}, k_7 із раундів 1,2 та 3, тим самим отримавши місце для розміщення ключових слів раундів 8,9 та 10 на їх місці. Це видалення виконується з використанням S-блоків в оберненому порядку після повернення ключів до їх k_{4n} значень. Це еквівалентно методу «зіг-загу» використаному в [27] для видалення раундів, але зараз він використовуєтсья для видалення ключів. Це заощаджує на 96 кубітів більше ніж в [27] і на 64 кубіти більше ніж в [30], де видалялось лише одне ключове слово. Оскільки кожне слово використовує чотири S-блоки, видалення потрібне для використання 12-ти додаткових S-блоків і значного заощадження кубітів. Видалення ключового слова k_{15} може бути виконано під час видалення п'ятого раунду без додаткової глибини. Точно так само видалення слова k_7 може бути виконано під час побудови восьмого раунду без додаткової глибини. Таким чином, глибина S-блоку для розширення ключа рівна двом, що включає в себе обчислення k_4 і видалення k_{11}, обидва з глибиною рівною одиниці. Якщо в майбутньому виявиться, що заощадження в кубітах не коштує додаткових вентилів і глибини S-блоку, то можна це ігнорувати і використовувати додаткові кубіти. Загальна глибина схеми використовує 46 S-блоків, 15 обчислень MixColumns, глибина кожного рівна 39-ти і глибиною 142 для використання AddRoundKey для кожного раунду.

Оптимізація для AES-192.

AES-192 трохи відрізняється в генерації ключів. AES-192 використовує S-блок тільки для кожного шостого ключа, і, оскільки для кожного ранду потрібно тільки чотири ключа, для деяких раундів потрібно тільки 16 S-блоків для повного обчислення, а деяким потрібно 16 і чотири додаткових для генерації ключа. Таким чином, хоча кількість раундів більша ніж в AES-128, генерується менше ключових слів. До того часу, коли необхідно обчислити k_{48}, таким чином k_{11} може бути замінений на k_6 і видалений з використанням оберненого S-блоку, таким чином зберігається 32 кубіти для чотирьох додаткових S-блоків.

Крім того, метод «зіг-загу» використаний в [27], використовував те саме число кубітів для AES-256, що і для AES-192. Це означає, що є місце для додаткових раундів або заощадження місця. Хоча не було зменшено кількість кубітів для генерації раундів, було використано частину цього додаткового пространства для розширення ключа. Замість того, щоб помістити 12-й раунд в 128 кубітів, що залишились, можна обернути частину 10-го раунду і повторно використати ці кубіти для зберігання частини 12-го раунду, таким чином отримавши достатню кількість кубітів для зберігання ключів раунду. Таким чином, коли генерується ключове слово k_{42}, воно генерується нижче того місця, де зберігається 11-й раунд, тим самим заощаджуючи ще 32 кубіти для витрат на ще чотирьох обернених S-блоках. В цілому, було збережено 64 кубіти, порівняно з [27]. Загальна глибина цієї схеми використовує 41 S-блок, 18 MixColumns і глибину 208, щоб застосувати AddRoundKey до кожного раунду.

Оптимізація для AES-256.

Для AES-256 можна використати ті самі методи, що і для AES-128, але використовується більше ключів, тому не можна їх так просто видалити. Після того, як 11-й раунд і k_{47} було побудовано, то ключі для 3-го і 7-го раундів (k_{11} і k_15) можуть бути видалені, таким чином, звільниться місце для ключів 12-го та 13-го раундів (k_{51} і k_{55}). Після 13-го раунду ключ k_{23} може бути видалений і на його місце можна записати ключ 14-го раунду (k_{59}). Загалом, це зберігає 96 кубітів. Загальна глибина

цієї схеми використовує 54 S-блоки, 22 MixColumns і глибину 267, щоб використати AddRoundKey до кожного раунду.

Цей метод визначення ключових слів під час генерації раунду і зберігання тільки k_{4n+3} для AES-128 і k_{4n+5} для AES-192 означає, що ключі між k_{4n} і цим ключем потрібно визначати кілька разів. Цей метод можна співставити з іншими методами отримання додактових ключів безпосередньо в раундах.

В таблиці 2.7 показано, яка кількість ресурсів потрібна для реалізації AES в [28].

Таблиця 2.7 – Кількість квантових ресурсів, необхідних на реалізацію AES.

	NOT	CNOT	Toffoli	Глибина S-блоку	Глибина Toffoli	Кубітів
AES-128	1.507	107.960	16.940	47	1.880	864
AES-192	1.692	125.580	19.580	41	1.640	896
AES-256	1.992	151.011	23.760	54	2.160	1.232

Для порівняння, в таблиці 2.8 вказано кількість небхідних ресурсів в [27] та [30].

Таблиця 2.8 – Кількість квантових ресурсів, необхідних на реалізацію AES в [27] та [30].

	[27]					[30]				
	NOT	CNOT	Toffoli	Глибина Toffoli	Кубітів	NOT	CNOT	Toffoli	Глибина Toffoli	Кубітів
AES-128	1.456	166.548	151.552	12.672	984	1.370	192.832	150.528	–	976
AES-192	1.608	189.432	172.032	11.088	1.112	–	–	–	–	–
AES-256	1.943	233.836	215.040	14.976	1.336	–	–	–	–	–

Порівнюючи ці дві таблиці, видно, що переглянутий дизайн S-блоку разом зі змінами в обробці ключів вагомо оптимізують кількість необхідних ресурсів.

Висновки до розділу 2

В цьому розділі було проведено аналіз джерел, котрі містять інформацію про витрати ресурсів на реалізацію AES та застосування до нього алгоритму Гровера. Також було отримано інформацію про те, як можна оптимізувати кількість необхідних ресурсів.

А саме, на реалізацію AES в квантовій моделі обчислень необхідно: 1.060.864 T-вентилі і 1.380.420 вентилів Кліффорда для AES-128; 1.204.224 T-вентилі і 1.567.296 вентилів Кліффорда для AES-192; 1.505.280 T-вентилів і 1.956.099 вентилів Кліффорда для AES-256 та 984, 1.112 і 1.336 кубітів, для $k = 128, 192, 256$ відповідно. Та в Таблиці 2.6 наведено кількість ресурсів, котра необхідна для застосування алгоритму Гровера до AES.

3 ОЦІНКА РЕСУРСОЄМНОСТІ РЕАЛІЗАЦІЇ ШИФРУ КАЛИНА В КВАНТОВІЙ МОДЕЛІ ОБЧИСЛЕНЬ

В даному розділі буде обчислено кількість вентилів необхідних для реалізації шифру Калина в квантовій моделі обчислень. Буде отримано оцінку необхідних квантових ресурсів для операції $AddRoundKey$, виконання підстановок шифру та для раундів шифру. Також буде описано алгоритм, за допомогою якого можна оптимізувати витрати квантових ресурсів.

3.1 Оцінка необхідних квантових вентилів необхідних для релізації S-блоків шифру Калина

Вентилі Тоффолі універсальні, це означає, що для будь-якої булевої функції $f(x_1,x_2,...,x_m)$ існує схема, яка складається з них. Вона приймає на вхід $x_1,x_2,...,x_m$ та деякі додаткові біти, 0 або 1, і виводить $x_1,x_2,...,x_m$, $f(x_1,x_2,...,x_m)$ та додаткові біти (котрі вважаються сміттям). Це означає, що можна використовувати вентилі Тоффолі для побудови систем, котрі будуть виконувати будь-які бажані обчислення булевих функцій оборотним чином, тобто будь-яка булева функція може бути побудована лише за допомогою вентилів Тоффолі.

Булефі функції зазвичай будують за допомогою вентилів OR, AND, і NOT, котрі можуть бути об'єднані для формування будь-якої БФ. Відомо, що те саме можна зробити тільки вентилями NOR (NOT-OR) або NAND (NOT-AND).

Вентиль Тоффолі може бути представлений як

$$Toffoli(a,b,c) = \begin{cases} (a,b,\overline{c}), \text{ при a=b=1;} \\ (a,b,c), \text{ в інших випадках.} \end{cases}$$

Оскільки перший та другий входи завжди дорівнюють першому та другому виходам ми можемо їх не враховувати, тоді маємо

$$Toffoli'(a,b,c) = \begin{cases} \overline{c}, \text{ при a=b=1;} \\ c, \text{ в інших випадках.} \end{cases}$$

При цьому можна задати вентиль NAND наступним чином:

$$NAND(a,b) = Toffoli'(a,b,c).$$

Це також доводить те, що вентилі Тоффолі універсальні, оскільки NAND є універсальними. Таблиці істинності для вентилів NAND, NOR та Тоффолі наведені в таблицях 3.1, 3.2 та 3.3 відповідно, а графічне представлення на рисунку 3.1.

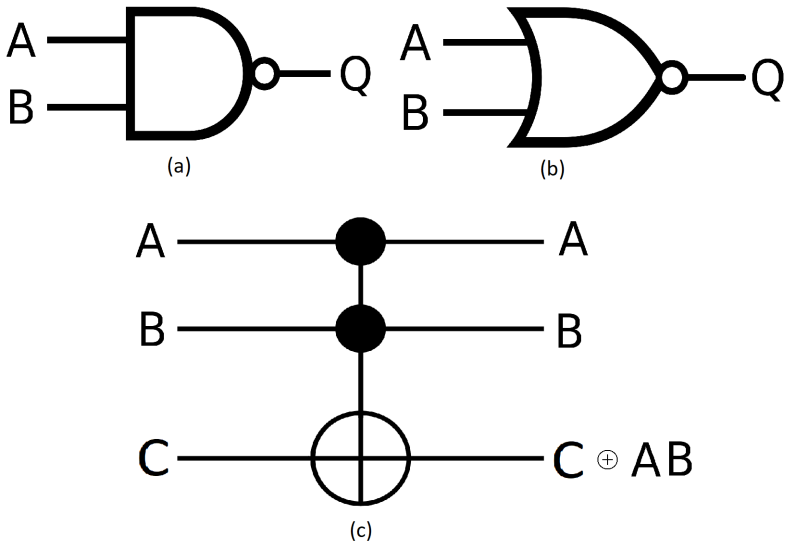

Рисунок 3.1 – Графічне представлення вентилів NAND(a), NOR(b) і Toffoli(c).

Враховуючи це, можна зрозуміти, що для виконання NAND в квантовій моделі обчислень необхіден один вентиль Toffoli та один кубіт для виходу. А для NOR(a,b) - ¬Toffoli(a,b,1), тобто для реалізації операції NOR в квантовій моделі обчислень необхідно використати один вентиль Toffoli, один кубіт для виходу та один вентиль Pauli-X до третього виходу вентилю Тоффолі (котрий виконує операцію NOT в квантовій моделі

Таблиця 3.1 – Таблиця істинності вентилю NAND.

Вхід		Вихід
A	B	A NAND B
0	0	1
0	1	1
1	0	1
1	1	0

Таблиця 3.2 – Таблиця істинності вентилю NOR.

Вхід		Вихід
A	B	A NAND B
0	0	1
0	1	0
1	0	0
1	1	0

Таблиця 3.3 – Таблиця істинності вентилю Тоффолі.

Вхід			Вихід		
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	1	1
1	1	1	1	1	0

обчислень).

Для всих чотирьх підстановок шифру Калина було сформовано таблиці істинності(табл. 3.6) та проведено перехід від ДНФ до СДНФ, далі за допомогою методу Квайна-Мак-Класкі було виконано мінімізацію отриманої булефої функції. Отриманий результат наведено у вигляді таблиць істинності в Додатку А.

Метод Квайна-МакКласкі для мінімізації булевих функцій, отриманих з S-блоків. Функція представлена в вигляді ДНФ. Тоді, щоб привести її до скороченої форми, необхідно виконувати операції склеювання та поглинання. Операція склеювання виконується при наявності пар членів, відповідаючих формі $w \bullet x$ або $w \bullet \bar{x}$, де w може бути кон'юнкією будь-якого числа змінних, що відрізняються від x. Після того, як всі такі пари будуть знайдені, проводиться операція склеювання: $w \bullet x \vee w \bullet \bar{x} = w \bullet (x \vee \bar{x}) = w$. Результати w тепер виступають додаковими членами.

Коли всі можливі склеювання були проведені, наступає черга операції поглинання $w \vee w \bullet y = w(1 \vee y) = w$, де w поглинає $w \bullet y$. Можливо чередування цих двох операцій, але їх необхідно виконувати до тих пір, поки це не буде неможливо. Після того, як неможливо буде з'єднати жодну з імплікант, отримана ДНФ являється скороченою.

Було побудовано схеми з вентилями NAND та NOR([31]), отримана оцінка необхідної кількості наведена в таблиці 3.4.

Таблиця 3.4 – Оцінка кількості вентилів, необхідних для виконання перестановок шифру Калина.

Підстановка	К-сть NAND	К-сть NOR
π_0	1278	1279
π_1	1273	1274
π_2	1252	1253
π_3	1262	1263

Розглянемо побудову схем за допомогою вентилів NAND. Кількість необхідних вентилів склала 1278, 1273, 1252 та 1262 для підстановок π_0, π_1, π_2 та π_3 відповідно. Для порівняння, різних дослідженнях було отримано кількості небхідних вентилів Тоффолі для реалізації S-блоку шифру AES(таблиця 3.5).

Якщо ж розглядати схеми, побудовані за допомогою вентилів NOR, то отримаємо, що необхідно 1279 вентилів Тоффолі та 1279 вентилів Pauli-X для підстановки π_0, 1274 вентилі Тоффолі та 1274 вентилі Pauli-X для π_1, 1253 вентилі Тоффолі та 1253 вентилі Pauli-X дляπ_2 та 1263 вентилі Тоффолі та 1263 вентилі Pauli-X для π_3.

3.2 Оцінка необхідної кількості квантових вентилів для реалізації шифру Калина

Існує 5 різних варіацій шифру Калина(таблиця 1.2), і три можливих кількості раундів(10, 14 або 18). З отриманих результатів, можна грубо оцінити кількість квантових вентилів, необхідних для реалізацаїї шифру Калина в квантовій моделі обчислень. Нехай, раунди не потребують ресурсів ні на що, окрім підстановок та операції $AddRoundKey$. Виконання операції $AddRoundKey$ двічі потребує кількість ресурів, приблизно рівну кількості вентилів, необхідних на виконання п'яти раундів.

Розглянемо результати, отримані за допомогою вентилів NAND.

Таблиця 3.5 – Кількість необхідних вентилів Тоффолі для реалізації S-блоку шифру AES.

Кількість вентилів Тоффолі	Джерело
512	[27]
1385	[27]
448	[30]

Таблиця 3.6 – Підстановка π_0.

i	x_7	x_6	x_5	x_4	x_3	x_2	x_1	x_0	s_7	s_6	s_5	s_4	s_3	s_2	s_1	s_0	
0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0	A8
1	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1	1	43
2	0	0	0	0	0	0	1	0	0	1	0	1	1	1	1	1	5F
3	0	0	0	0	0	0	1	1	0	0	0	0	0	1	1	0	06
4	0	0	0	0	0	1	0	0	0	1	1	0	1	0	1	1	6B
5	0	0	0	0	0	1	0	1	0	1	1	1	0	1	0	1	75
6	0	0	0	0	0	1	1	0	0	1	1	0	1	1	0	0	6C
7	0	0	0	0	0	1	1	1	0	1	0	1	1	0	0	1	59
8	0	0	0	0	1	0	0	0	0	1	1	1	0	0	0	1	71
9	0	0	0	0	1	0	0	1	1	1	0	1	1	1	1	1	DF
10	0	0	0	0	1	0	1	0	1	0	0	0	0	1	1	1	87
11	0	0	0	0	1	0	1	1	1	0	0	1	0	1	0	1	95
12	0	0	0	0	1	1	0	0	0	0	0	1	0	1	1	1	17
13	0	0	0	0	1	1	0	1	1	1	1	1	0	0	0	0	F0
14	0	0	0	0	1	1	1	0	1	1	0	1	1	0	0	0	D8
15	0	0	0	0	1	1	1	1	0	0	0	0	1	0	0	1	09
16	0	0	0	1	0	0	0	0	0	1	1	0	1	1	0	1	6D
...	...								...								...

Враховуючи це, можна грубо оцінити кількість вентилів, необхідну на реалізацію одного раунду AES (не враховуючи операцію $AddRoundKey$). Було отримано, що для одного раунду необхідно $x_0 + x_1 + x_2 + x_3 = N_T$, де x_0,x_1,x_2,x_3 — кількість вентилів Тоффолі, необхідних для підстановок π_0, π_1, π_2 та π_3 відповідно, а N_T — загальна кількість вентилів, необхідна для реалізації одного раунду шифру Калина (не враховуючи операцію $AddRoundKey$). Отже, для одного рануду необхідно $1.278 + 1.273 + 1.252 + 1.262 = 5.065$ вентилів Тоффолі.

Калина-128/128. Спочатку один раз виконується операція $AddRoundKey$, потім десять раундів, в кожному з яких по одному разу виконується операція $AddRoundKey$ та чотири перестановки (π_0, π_1, π_2 та π_3). Нехай, на всі виконання операції $AddRoundKey$, а їх буде 11 (на початку, та в кожному раунді), потрібна така ж кількість вентилів, як і

на виконання 30-и раундів. Тоді, для реалізації шифру Калина-128/128 необхідно $(30 + 10) \cdot 5.065 = 202.600$ вентилів Тоффолі.

Калина-256/256. Спочатку один раз виконується операція $AddRoundKey$, потім чотирнадцять раундів, в кожному з яких по одному разу виконується операція $AddRoundKey$ та чотири перестановки (π_0, π_1, π_2 та π_3). Нехай, на всі виконання операції $AddRoundKey$, а їх буде 15 (на початку, та в кожному раунді), потрібна така ж кількість вентилів, як і на виконання 40-а раундів. Тоді, для реалізації шифру Калина-256/256 необхідно $(40 + 14) \cdot 5.065 = 273.510$ вентилів Тоффолі.

Калина-512/512. Спочатку один раз виконується операція $AddRoundKey$, потім вісімнадцять раундів, в кожному з яких по одному разу виконується операція $AddRoundKey$ та чотири перестановки (π_0, π_1, π_2 та π_3). Нехай, на всі виконання операції $AddRoundKey$, а їх буде 19 (на початку, та в кожному раунді), потрібна така ж кількість вентилів, як і на виконання 50-и раундів. Тоді, для реалізації шифру Калина-512/512 необхідно $(50 + 18) \cdot 5.065 = 344.420$ вентилів Тоффолі.

Вентилі Тоффолі — універсальні в класичній моделі обчислень. В квантовій моделі обчислень повний базис — T-вентилі та вентилі Кліффорда, генератори NOT і CNOT. Один вентиль Тоффолі = 7 вентилям Тоффолі та 8 вентилям Кліффорда. Отже, для реалізації шифру Калина-128/128 в квантовій моделі обчислень, необхідно $202.600 \cdot 7 = 1.418.200$ T-вентилі, $202.600 \cdot 8 = 1.620.800$ вентилів Кліффорда та 1225 кубітів, для Калина-256/256 — $273.510 \cdot 7 = 1.914.570$ T-вентилі, $273.510 \cdot 8 = 2.188.080$ вентилів Кліффорда та 1583 кубіти, для Калина-512/512 — $344.420 \cdot 7 = 2.410.940$ T-вентилі,

Таблиця 3.7 – Оцінка, отримана при використанні вентилів NAND

	Розмір блоку	Довжина ключа	К-сть раундів	К-сть вентилів Тоффолі
1	128	128	10	202.600
2	256	256	14	273.510
3	512	512	18	344.420

$344.420 \cdot 8 = 2.755.360$ вентилів Кліффорда та 2356 кубітів.(табл. 3.8)

Розглянемо результати, отримані за допомогою вентилів NOR. Можна грубо оцінити кількість вентилів, необхідну на реалізацію одного раунду AES (не враховуючи операцію $AddRoundKey$). Було отримано, що для одного раунду необхідно $x_0 + x_1 + x_2 + x_3 = N_T$, де x_0,x_1,x_2,x_3 — кількість вентилів Тоффолі, необхідних для перестановок π_0, π_1, π_2 та π_3 відповідно, а N_T — загальна кількість вентилів Тоффолі, та $y_0 + y_1 + y_2 + y_3 = N_P$, де y_0,y_1,y_2,y_3 — кількість вентилів Pauli-X, необхідних для перестановок π_0, π_1, π_2 та π_3, а N_P — загальна кількість вентилів Pauli-X, необхідна для реалізації одного раунду шифру Калина (не враховуючи операцію $AddRoundKey$). Отже, для одного рануду необхідно $1.279 + 1.274 + 1.253 + 1.263 = 5.069$ вентилів Тоффолі та $1.279 + 1.274 + 1.253 + 1.263 = 5.069$ вентилів Pauli-X. Далі все майже так, як і при обчисленні першої оцінки.

Калина-128/128. Для реалізації шифру Калина-128/128 необхідно $(30 + 10) \cdot 5.069 = 202.760$ вентилів Тоффолі та $(30 + 10) \cdot 5.069 = 202.760$ вентилів Pauli-X. Або $(30 + 10) \cdot 5.069 = 202.760$ вентилів Pauli-X, $202.760 \cdot 7 = 1.419.320$ T-вентилів та $202.760 \cdot 8 = 1.622.080$ вентилів Кліффорда.

Калина-256/256. Для реалізації шифру Калина-256/256 необхідно $(40 + 14) \cdot 5.069 = 273.726$ вентилів Тоффолі та $(40 + 14) \cdot 5.069 = 273.726$ вентилів Pauli-X. Або $(40 + 14) \cdot 5.069 = 273.726$ вентилів Pauli-X, $273.726 \cdot 7 = 1.916.082$ T-вентилів та $273.726 \cdot 8 = 2.189.808$ вентилів Кліффорда.

Таблиця 3.8 – Кількість ресурсів, необхідних для реалізації шифру Калина в квантовій моделі обчислень.

	К-сть Т-вентилів	К-сть вентилів Кліффорда	К-сть кубітів
Калина-128/128	1.418.200	1.620.800	1225
Калина-256/256	1.914.57	2.188.080	1583
Калина-512/512	2.410.940	2.755.360	2356

Калина-512/512. Для реалізації шифру Калина-512/512 необхідно $(50 + 18) \cdot 5.069 = 344.692$ вентилів Тоффолі та $(50 + 18) \cdot 5.069 = 344.692$ вентилів Pauli-X. Або $(50 + 18) \cdot 5.069 = 344.692$ вентилів Pauli-X, $344.692 \cdot 7 = 2.412.844$ T-вентилів та $344.692 \cdot 8 = 2.757.536$ вентилів Кліффорда.

3.3 Оптимізація кількості квантових вентилів, необхідних на реалізацію S-блоків

Спосіб заключається в тому, що спочатку зменшується мультиплікативна складність, а далі оптимізуються лінійні компоненти, що приводить до менших схем [29].

Перший крок

Перший крок заключається в ідентифікації нелінійних компонентів підсхеми, котрі повинні бути оптимізвані і зменшується кількість вентилів AND. Це зробити не дуже легко. Наприклад, на рисунку 3.2 наведено дві схеми, які рахують одну й ту саму функцію. Але не очевидно, як алгоритмічно трансформувати одну в іншу.

Пошук схем з мінімальною мультиплікативною складністю, вірогідно, являється дуже складною задачею. Тим не менше існують методи редукції, котрі на практиці дають схеми з невеликою, і часто оптимальною мультиплікативною складністю. Ці методи орієнтовані виключно на симетричні функції (ті, чиє значення залежить тільки від ваги Хеммінга на вході).

Архітектура башенних полів для інверсії в $GF(2^8)$ має легко

Таблиця 3.9 – Оцінка, отримана при використанні вентилів NOR

	Розмір блоку	Довжина ключа	К-сть раундів	К-сть вент. Тоффолі	К-сть вент. Pauli-X
1	128	128	10	202.760	202.760
2	256	256	14	273.726	273.726
3	512	512	18	344.692	344.692

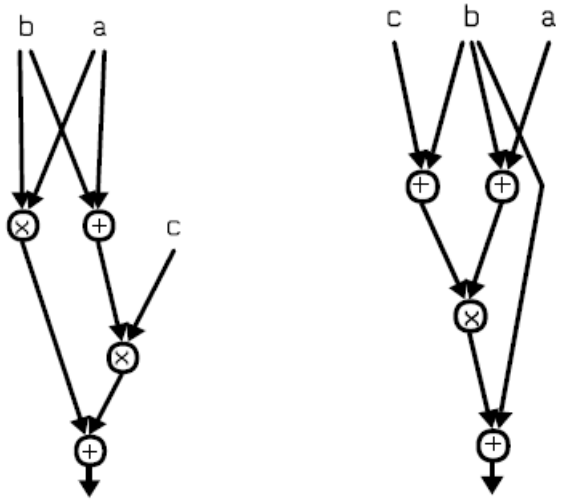

Рисунок 3.2 – Різні схеми для обчислення однієї функції.

ідентифіковані нелінійні компоненти, котрі відповідають інверсії в підполях. Перший крок заключається в тому, щоб зосередитись на одному з цих компонентів і отримати схему, котра використовує кілька $\wedge$ вентилів. Компонент для інверсії в полі $GF(2^2)$ замалий, щоб можна було його значно оптимізовати, тому необхідно зосереджуватись на інверсії в полі $GF(2^4)$.

Забагато представлень в $GF(2^4)$. Будуємо

- $GF(2^2)$ приєднуючи корінь W з $x^2 + x + 1$ над $GF(2)$;
- $GF(2^4)$ приєднуючи коріть Z з $x^2 + x + W^2$ над $GF(2^2)$.

Представляємо $GF(2^2)$, використовуючи базис (W,W^2) і $GF(2^4)$ використовуючи базис (Z^2,Z^8). Таким чином, елемент $\delta \in GF(2)$ записується як $\delta_1 Z^2 + \delta_2 Z^8$, де $\delta_1,\delta_2 \in GF(2^2)$. Точно так само елемент γ в полі $GF(2^2)$ записується як $\gamma_1 W + \gamma_2 W^2$, де $\gamma_1,\gamma_2 \in GF(2)$. Оскільки Z задовільняє $x^2 + x + W^2 = 0$ і W задовільняє $x^2 + x + 1 = 0$, можна порахувати, що $Z^4 = Z^2 + W$, $Z^8 = Z^2 + 1$, $Z^{10} = Z^4 + Z^2 = W$, $Z^{16} = Z^8 + W$, $W^3 = W^2 + W$, $W^4 = W$, і $W^5 = W^2$. Ці рівняння можуть бути використані для зменшення виразів для перевірки рівностей.

Використовуючи це представлення, елемент в полі $GF(2^4)$ можна записати як $\Delta = (x_1 W + x_2 W^2)Z^2 + (x_3 W + x_4 W^2)Z^8$, де $x_1,x_2,x_3,x_4 \in GF(2)$. Інверсія цього елемента може бути придставлена як

$\Delta' = (y_1W + y_2W^2)Z^2 + (y_3W + y_4W^2)Z^8$, далі можна рахувати використовуючи наступні поліноми над $GF(2)$:

- $y_1 = x_2x_3x_4 + x_1x_3 + x_2x_3 + x_3 + x_4$
- $y_2 = x_1x_3x_4 + x_1x_3 + x_2x_3 + x_2x_4 + x_4$
- $y_3 = x_1x_2x_4 + x_1x_3 + x_1x_4 + x_1 + x_2$
- $y_4 = x_1x_2x_3 + x_1x_3 + x_1x_4 + x_2x_4 + x_2$

Символічний результат $(QW + QW^2)Z^2 + (QW + QW^2)Z^8$, де $Q = x_1x_2x_3x_4 + x_1x_2x_3 + x_1x_2x_4 + x_1x_3x_4 + x_2$ $x_3x_4 + x_1x_2 + x_1x_3 + x_1x_4 + x_2x_3 + x_2x_4 + x_3x_4 + x_1 + x_2 + x_3 + x_4$. Той факт, що значення Q рівне 1, якщо всі чотири змінні не мають значення 0, коли воно рівне 0, можна побачити, помітивши те, що функція $\sum_4^4 + \sum_3^4 + \sum_2^4 + \sum_1^4$ симетрична. Якщо задано рівно чотири змінні, то перший член дає значення 1 (а інші - 0); якщо задані три, то другий, третій та четвертий члени дають значення 1; ящо задано рівно два, то тільки третій дає 1; і якщо заданий тільки один, то тільки останній дає 1. Відповідно, результат рівний 1, за виключенням нульового входу.

Таким чином, задача, поставлена перед нами, складається в тому, щоб побудувати схему з чотирьма входами і чотирьма виходами, котра порахує вищевказану систему рівнянь з використанням як можна меншої кількості $\wedge$ вентилів. В наш час, програми еврістичного пошуку можуть оброблювати функції які мають один вхід, та не більше восьми виходів. Це означає, що можна побудувати оптимальні схеми для кожного з чотирьох рівнянь окремо, але не для всієї системи. Для повної системи необхіден наступний підхід:

- вибрати рівняння, та побудувати для нього ефективну схему;
- зберегти проміжні результати функцій, пораховані на попередніх етапах, для можливого викристання при побудові схеми для наступного рівняння, котре має бути вирішено;
- повторювати, поки всі рівняння не будуть вирішені.

Перший крок нетривіальний навіть для предикатів на кількох входах. Використовувана евристика заснована на методах автоматичного

доведення теорем. Виявляється, що три множення достатньо для обчислення будь-якого предикату по чотирьом змінним.

Було виконано всі кроки, описані вище для кожного з 24 порядків $y_1; y_2; y_3; y_4$. Порядок $(y_4; y_2; y_1; y_3)$ дав найкращий результат. Результуюча схема, в вигляді $straight-line$ програми над $GF(2)$, представлена в таблиці 3.10 (виходи позначено (*)).

Ця схема налічує 5 вентилів $\wedge$ і 11 вентилів $\oplus$. Мультиплікативна складність функції - це число $GF(2)$ множень, необхідних і достатніх для обчислення. При заданому представленні для $GF(2^4)$ мультиплікативна складність інверсії рівна 5. Всі чотири виходa, котрі повинні бути пораховані мають степінь 3. Для обчислення многочлена степені 2 потрібен один вентиль $\wedge$. Далі необхіден додатковий вентиль $\wedge$, щоб отримати кожен з чотирьох лінійно незалежних многочленів, оскільки кожен степені 3.

Другий крок

Другий крок заключається в тому, що необхідно знайти максимальні лінійні компоненти схеми і після мінімізувати кількість вентилів XOR, необхідних для знаходження цільових функцій, порахованих в цих лінійних компонентах.

Лінійна $straight - line$ програма над полем F, це варіація, котра не допускає множення змінних. Тобто, кожен рядок програми має вигляд $u := \lambda v + \mu w$, де λ, μ в F, а v, v - змінні. Побудова лінійної схеми для f еквівалентно побудові лінійної програми над $GF(2)$, котра обчислює f (слід звернути увагу, що над $GF(2)$ λ і μ завжди рівні 1, і, відповідно,

Таблиця 3.10 – Інверсія в $GF(2^4)$

$t_1 = x_1 + x_2$	$t_2 = x_1 \times x_3$	$t_3 = x_4 + t_2$
$t_4 = t_1 \times t_3$	$y_4 = x_2 + t_4$ (*)	$t_5 = x_3 + x_4$
$t_6 = x_2 + t_2$	$t_7 = t_6 \times t_5$	$y_2 = x_4 + t_7$(*)
$t_8 = x_3 + y_2$	$t_9 = t_3 + y_2$	$t_{10} = x_4 + t_9$
$y_1 = t_{10} + t_8$ (*)	$t_{11} = t_3 + t_{10}$	$t_{12} = y_4 \times t_{11}$
$y_3 = t_{12} + t_1$ (*)		

николи не пишуться явно).

Лінійна програма над $GF(2)$ називається вільною від відміни, якщо для кожного рядка програми $u := u + w$ жодна зі змінних виразу для v не фігурує в виразі для w, тобто в обчисленні немає відміни змінних.

Багато авторів вважають, що завжди існує оптимальна лінійна програма без відміни над $GF(2)$. Невеликий приклад, котрий показує що це не так:

$$x_1 + x_2; x_1 + x_2 + x_3; x_1 + x_2 + x_3 + x_4; x_2 + x_3 + x_4.$$

Можна побачити, що оптимальна прямолінійна програма без відміни має довжину 5. Рішення довжини 4, котре допускає відміни виглядає так:

$$v_1 = x_1 + x_2; v_2 = v_1 + x_3; v_3 = v_2 + x_4; v_4 = v_3 + x_1.$$

Даний алгоритм допоможе оптимізувати отриману оцінку в подальшій роботі.

Висновки до розділу 3

В даному розділі було отримано оцінку кількості квантових ресурсів, а саме — вентилів Тоффолі, необхідних для реалізації шифру Калина в квантовій моделі обчислень. Описаний алгоритм, котрий допоможе знизити затрати при реалізації шифру Калина в квантовій моделі обчислень.

Було виявлено, що для реалізації шифру Калина-128/128 необхідно близько $202.600 \cdot 7 = 1.418.200$ T-вентилі, $202.600 \cdot 8 = 1.620.800$ вентилів Кліффорда та 1225 кубітів, для Калина-256/256 — $273.510 \cdot 7 = 1.914.570$ T-вентилі, $273.510 \cdot 8 = 2.188.080$ вентилів Кліффорда та 1583 кубіти, для Калина-512/512 — $344.420 \cdot 7 = 2.410.940$ T-вентилі, $344.420 \cdot 8 = 2.755.360$ вентилів Кліффорда та 2356 кубітів.

ВИСНОВКИ

Під час цієї роботи було проаналізовано та стисло описано джерела необхідні для дослідження, а саме, джерела, в котрих було описано міжнародний стандарт шифрування AES, національний стандарт України ДСТУ 7624:2014 (шифр Калина), деякі атаки на ці шифри, алгоритм Гровера та алгоритми, за допомогою яких можна зменшити витрати квантових ресурсів, необхідних на реалізацію криптографічних алгоритмів в квантовій моделі обчислень. Описано особливості роботи шифрів AES та Калини.

Було досліджено методи оцінки необхідних квантових ресурсів на реалізацію криптографічних примітивів в квантовій моделі обчислень, досліджені самі оцінки, та методи за допомогою яких затрати зменшували.

Було проведено оцінку кількості вентилів, необхідних на реалізацію окремих складових шифру Калина, а саме операції AddRoundKey та раундів шифру. Отримано, що для одного виконання операції AddRoundKey необхідно близько 12.737 вентилів Тоффолі, а для одного раунду шифрування — близько 17.832 вентилів Тоффолі. Для реалізації підстановок шифру Калина необхідно 1278, 1273, 1252 та 1262 вентилів Тоффолі, для підстановок π_0, π_1, π_2 та π_3 віповідно.

Проведено повну оцінку кількості квантових ресурсів, а саме - кількість кубітів, вентилів T та Кліффорда, необхідних для реалізації шифру Калина в квантовій моделі обчислень. Отримано, що для Калина-k/k необхідно близько $202.600 \cdot 7 = 1.418.200$ T-вентилі, $202.600 \cdot 8 = 1.620.800$ вентилів Кліффорда та 1225 кубітів для $k = 128$; $273.510 \cdot 7 = 1.914.570$ T-вентилі, $273.510 \cdot 8 = 2.188.080$ вентилів Кліффорда та 1583 кубіти для $k = 256$; $344.420 \cdot 7 = 2.410.940$ T-вентилі, $344.420 \cdot 8 = 2.755.360$ вентилів Кліффорда та 2356 кубітів для $k = 512$. Оскільки шифр Калина дуже схожий з шифром AES, можна порівняти

оцінки, отримані для AES, та оцінку, отриману в цій роботі. Для AES було отримано: 1.060.864 T-вентилі, 1.380.420 вентилів Кліффорда та 984 кубітів для AES-128; 1.204.224 T-вентилі, 1.567.296 вентилів Кліффорда та 1112 кубітів для AES-192; 1.505.280 T-вентилів, 1.956.099 вентилів Кліффорда та 1336 кубітів для AES-256. На підставі цього отримані результати можна вважати правильними, оскільки оцінка проводилась для підстановок, заданих авторами стандарту, а при обчисленні затрат на реалізацію AES використовувались математично підібрані таблиці підстановок. Оскільки існує максимально 72-кубітний квантовий комп'ютер, то практична реалізація шифру Калина на ньому неможлива.

У подальшій роботі планується за допомогою алгоритмів, описаних в дослідженні, мінімізувати кількість необхідних квантових ресурсів на реалізацію шифру Калина в квантовій моделі обчислень.

ПЕРЕЛІК ПОСИЛАНЬ

1. Системы обработки информации. Защита криптографическая. Алгоритмы криптографического преобразования: ДСТУ ГОСТ 28147:2009. – [Чинний від 2009-02-01]. – К.: Держспоживстандарт України, 2008. – 28 с. – (Національний стандарт України).

2. Takanori I. A Single-Key Attack on the Full GOST Block Cipher / I. Takanori // Fast Software Encryption 18th International Workshop (FSE-2011), Springer LNCS 6733, 2011. – pp. 290-305.

3. Обґрунтування вимог, побудування та аналіз перспективних симетричних криптоперетворень на основі блочних шифрів / [О. О. Кузнецов, Р. В. Олійников, Горбенко Ю. І. та ін.] // Вісн. Нац. ун-ту "Львів. політехніка". – 2014.

4. R. Oliynykov, O. Kazymyrov, O. Kachko et al. Source code for performance estimation of 64-bit optimized implementation of the block ciphers Kalyna, AES, GOST, BelT, Kuznyechik. 2015 [Electronic resource] // Mode of access: www. URL: https://github.com/Roman-Oliynykov/ciphers-speed/ (20.06.2017).

5. Я.Р. Совин, В.І. Отенко, Є.Ф. Штефанюк. Ефективна реалізація алгоритму БСШ ДСТУ 7624:2014 ("Калина") для 8/16/32-бітових вбудованих систем. 2017

6. ДСТУ 7624:2014 Інформаційні технології. Криптографічний захист інформації. Алгоритм симетричного блокового перетворення

7. Riham AlTawy, Ahmed Abdelkhalek, and Amr M. Youssef. A Meet-in-the-Middle Attack on Reduced-Round Kalyna-b/2b https://eprint.iacr.org/2015/762.pdf

8. Akshima, Donghoon Chang, Mohona Ghosh, Aarushi Goel, and Somitra Kumar Sanadhya. Single Key Recovery Attacks on 9-round Kalyna-128/256 and Kalyna-256/512. 2016.

9. [DR99] Joan Daemen and Vincent Rijmen. AES proposal: Rijndael.

1999.

10. Xavier Bonnetain, María Naya-Plasencia and André Schrottenloher. Quantum Security Analysis of AES

11. [GLRS16] Markus Grassl, Brandon Langenberg, Martin Roetteler, and Rainer Steinwandt. Applying Grover's Algorithm to AES: Quantum Resource Estimates. In PQCrypto, volume 9606 of Lecture Notes in Computer Science, pages 29–43. Springer, 2016.

12. [BLNS18] Christina Boura, Virginie Lallemand, María Naya-Plasencia, and Valentin Suder. Making the impossible possible. J. Cryptology, 31(1):101–133, 2018.

13. [BHT98] Gilles Brassard, Peter Høyer, and Alain Tapp. Quantum cryptanalysis of hash and claw-free functions. In LATIN, volume 1380 of Lecture Notes in Computer Science, pages 163–169. Springer, 1998.

14. [BNS14] Christina Boura, María Naya-Plasencia, and Valentin Suder. Scrutinizing and improving impossible differential attacks: Applications to CLEFIA, Camellia, LBlock and Simon. In Advances in Cryptology - ASIACRYPT 2014. Proceedings, Part I, volume 8873 of Lecture Notes in Computer Science, pages 179–199. Springer, 2014.

15. [DKR97] Joan Daemen, Lars R. Knudsen, and Vincent Rijmen. The block cipher square. In FSE, volume 1267 of Lecture Notes in Computer Science, pages 149–165. Springer, 1997.

16. [FKL+00] Niels Ferguson, John Kelsey, Stefan Lucks, Bruce Schneier, Michael Stay, David A. Wagner, and Doug Whiting. Improved cryptanalysis of Rijndael. In FSE, volume 1978 of Lecture Notes in Computer Science, pages 213–230. Springer, 2000.

17. [DFJ13] Patrick Derbez, Pierre-Alain Fouque, and Jérémy Jean. Improved key recovery attacks on reduced-round AES in the single-key setting. In Thomas Johansson and Phong Q. Nguyen, editors, Advances in Cryptology - EUROCRYPT 2013. Proceedings, volume 7881 of Lecture Notes in Computer Science, pages 371–387. Springer, 2013.

18. [BDD+12] Charles Bouillaguet, Patrick Derbez, Orr Dunkelman,

Pierre-Alain Fouque, Nathan Keller, and Vincent Rijmen. Low-data complexity attacks on AES. IEEE Trans. Information Theory, 58(11):7002–7017, 2012.

19. [DS08] Hüseyin Demirci and Ali Aydin Selçuk. A meet-in-the-middle attack on 8- round AES. In FSE, volume 5086 of Lecture Notes in Computer Science, pages 116–126. Springer, 2008.

20. Lov K. Grover. A fast quantum mechanical algorithm for database search. In Gary L. Miller, editor, Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing (STOC 1996), pages 212–219. ACM, 1996.

21. Michel Boyer, Gilles Brassard, Peter Høyer, and Alain Tapp. Tight bounds on quantum searching. Fortschritte der Physik, 46:493–506, 1998. arXiv:quant-ph/9605034.

22. Michael A. Nielsen and Issac L. Chuang. Quantum Computation and Quantum Information. Cambridge University Press, Cambridge, UK, 2000.

23. Adriano Barenco, Charles H. Bennett, Richard Cleve, David P. DiVincenzo, Norman Margolus, Peter W. Shor, Tycho Sleator, John Smolin, and HaraldWeinfurter. Elementary gates for quantum computation. Physical Review A, 52(5):3457–3467, 1995.

24. NIST. Specification for the ADVANCED ENCRYPTION STANDARD (AES). Federal Information Processing Standards Publication 197, November 2001.

25. Lov K. Grover. A fast quantum mechanical algorithm for database search. In Gary L. Miller, editor, Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing (STOC 1996), pages 212–219. ACM, 1996.

26. Shane Kepley and Rainer Steinwandt. Quantum circuits for F2n - multiplication with subquadratic gate count. Quantum Information Processing, 14(7):2373–2386, 2015.

27. Markus Grassl, Brandon Langenberg, Martin Roetteler, and Rainer Steinwandt. Applying Grover's Algorithm to AES: Quantum Resource Estimates. In Tsuyoshi Takagi, editor, Post-Quantum Cryptography PQCrypto

2016, volume 9606 of Lecture Notes in Computer Science, pages 29–43. Springer, 2016.

28. Brandon Langenberg, Hai Pham, and Rainer Steinwandt. Reducing the Cost of Implementing AES as a Quantum Circuit

29. Joan Boyar and Rene Peralta. A New Combinational Logic Minimization Technique with Applications to Cryptology. In Paola Festa, editor, International Symposium on Experimental Algorithms SEA 2010, volume 6049 of Lecture Notes in Computer Science, pages 178–189. Springer, 2010. Доступно за посиланням: https://eprint.iacr.org/2009/191. Notes in Computer Science, pages 29–43. Springer, 2016.

30. Mishal Almazrooie, Azman Samsudin, Rosni Abdullah, and Kussay N. Mutter. Quantum reversible circuit of AES-128. Quantum Information Processing, 17(5):112, 2018.

31. url: https://github.com/SaliiRV/MapGates

ДОДАТОК А ВЕЛИКІ РИСУНКИ ТА ТАБЛИЦІ

А.1 Мінімізована таблиця істинності першої підстановки

x7,x6,x5,x4,x3,x2,x1,x0,,s7,s6,s5,s4,s3,s2,s1,s0

1,X,1,1,X,0,1,X,,1,0,0,0,0,0,0,0,0

0,0,1,0,X,X,0,1,,1,0,0,0,0,0,0,0,0

1,X,0,1,0,X,0,1,,1,0,0,0,0,0,0,0,0

1,0,0,X,0,1,0,1,,1,0,0,0,0,0,0,0,0

0,1,0,X,0,0,1,0,,1,0,0,0,0,0,0,0,0

X,X,1,1,1,0,1,X,,1,0,0,0,0,0,0,0,0

1,1,1,0,X,1,0,X,,1,0,0,0,0,0,0,0,0

0,0,0,1,X,0,X,1,,1,0,0,0,0,0,0,0,0

0,0,1,X,1,X,1,1,,1,0,0,0,0,0,0,0,0

X,0,0,X,1,0,1,1,,1,0,0,0,0,0,0,0,0

1,0,1,1,X,X,0,1,,1,0,0,0,0,0,0,0,0

X,1,0,X,0,0,0,1,,1,0,0,0,0,0,0,0,0

0,0,1,X,0,1,X,0,,1,0,0,0,0,0,0,0,0

0,0,1,1,X,0,X,0,,1,0,0,0,0,0,0,0,0

0,X,1,0,0,X,1,0,,1,0,0,0,0,0,0,0,0

X,X,1,1,0,1,1,0,,1,0,0,0,0,0,0,0,0

X,1,X,1,1,0,0,0,,1,0,0,0,0,0,0,0,0

0,1,0,1,1,1,0,X,,1,0,0,0,0,0,0,0,0

1,0,1,0,0,X,1,1,,1,0,0,0,0,0,0,0,0

0,1,1,0,X,0,1,1,,1,0,0,0,0,0,0,0,0

0,1,X,0,0,1,0,1,,1,0,0,0,0,0,0,0,0

1,0,0,0,0,0,X,0,,1,0,0,0,0,0,0,0,0

1,1,1,0,0,X,0,0,,1,0,0,0,0,0,0,0,0

1,0,X,0,1,0,0,0,,1,0,0,0,0,0,0,0,0

X,0,0,X,1,0,0,X,,0,1,0,0,0,0,0,0,0

0,1,0,0,0,X,0,X,,0,1,0,0,0,0,0,0,0

X,0,0,1,1,0,X,1,,0,1,0,0,0,0,0,0,0

1,X,0,1,1,1,1,X,,1,0,0,0,0,0,0,0,0

0,0,X,0,1,X,0,1,,1,0,0,0,0,0,0,0,0

0,X,1,0,X,1,1,0,,1,0,0,0,0,0,0,0,0

1,1,0,0,0,X,1,0,,1,0,0,0,0,0,0,0,0

X,0,X,1,1,0,1,X,,1,0,0,0,0,0,0,0,0

X,0,1,X,X,1,1,0,,1,0,0,0,0,0,0,0,0

0,1,X,0,1,0,0,X,,1,0,0,0,0,0,0,0,0

1,1,1,X,1,X,1,1,,1,0,0,0,0,0,0,0,0

1,X,0,X,1,1,1,1,,1,0,0,0,0,0,0,0,0

1,X,X,1,1,0,1,1,,1,0,0,0,0,0,0,0,0

X,X,1,1,1,1,0,1,,1,0,0,0,0,0,0,0,0

1,0,1,1,X,1,X,0,,1,0,0,0,0,0,0,0,0

1,0,X,1,0,1,X,0,,1,0,0,0,0,0,0,0,0

X,1,0,0,1,0,X,0,,1,0,0,0,0,0,0,0,0

X,0,X,0,1,1,1,0,,1,0,0,0,0,0,0,0,0

X,1,0,0,1,X,0,0,,1,0,0,0,0,0,0,0,0

1,X,X,1,0,0,0,0,,1,0,0,0,0,0,0,0,0

1,1,0,0,0,0,X,1,,1,0,0,0,0,0,0,0,0

0,0,X,1,0,1,1,1,,1,0,0,0,0,0,0,0,0

0,1,1,1,X,1,0,1,,1,0,0,0,0,0,0,0,0

1,0,1,0,1,0,X,0,,1,0,0,0,0,0,0,0,0

0,0,0,0,1,X,1,0,,1,0,0,0,0,0,0,0,0

0,X,0,1,0,1,0,0,,1,0,0,0,0,0,0,0,0

0,X,0,0,0,0,0,0,,1,0,0,0,0,0,0,0,0

0,0,1,1,0,1,X,X,,0,1,0,0,0,0,0,0,0

X,0,0,0,0,1,0,X,,0,1,0,0,0,0,0,0,0

X,X,1,1,0,1,0,1,,0,1,0,0,0,0,0,0,0

1,X,1,1,1,X,1,0,,0,1,0,0,0,0,0,0

X,1,1,0,X,0,1,0,,0,1,0,0,0,0,0,0

1,0,X,1,1,0,0,1,,0,1,0,0,0,0,0,0

0,1,1,0,0,1,1,1,,0,1,0,0,0,0,0,0

0,X,0,X,X,0,0,1,,0,1,0,0,0,0,0,0

X,1,0,1,1,1,1,X,,0,1,0,0,0,0,0,0

1,1,1,0,X,1,0,X,,0,1,0,0,0,0,0,0

0,1,1,X,1,0,X,1,,0,1,0,0,0,0,0,0

X,0,1,0,1,X,1,1,,0,1,0,0,0,0,0,0

0,X,X,0,1,1,0,1,,0,1,0,0,0,0,0,0

1,1,X,X,0,0,0,1,,0,1,0,0,0,0,0,0

1,1,0,X,1,0,X,0,,0,1,0,0,0,0,0,0

1,1,1,X,X,1,0,0,,0,1,0,0,0,0,0,0

1,0,0,1,0,1,1,X,,0,1,0,0,0,0,0,0

0,1,1,1,1,X,1,1,,0,1,0,0,0,0,0,0

1,X,0,0,0,1,1,1,,0,1,0,0,0,0,0,0

1,1,0,1,X,1,0,1,,0,1,0,0,0,0,0,0

0,0,1,1,1,0,X,0,,0,1,0,0,0,0,0,0

X,1,0,0,0,1,1,0,,0,1,0,0,0,0,0,0

1,X,0,0,0,0,1,0,,0,1,0,0,0,0,0,0

0,0,1,X,1,1,0,0,,0,1,0,0,0,0,0,0

X,0,1,1,X,1,X,1,,0,0,1,0,0,0,0,0

0,1,1,X,0,X,X,0,,0,0,1,0,0,0,0,0

1,0,1,1,0,X,0,X,,0,0,1,0,0,0,0,0

0,X,0,0,X,1,0,1,,0,0,1,0,0,0,0,0

0,X,1,0,0,0,X,0,,0,0,1,0,0,0,0,0

0,0,0,X,X,0,0,0,,0,0,1,0,0,0,0,0

X,1,0,0,0,1,1,1,,0,0,1,0,0,0,0,0

1,1,X,1,1,1,0,0,,0,0,1,0,0,0,0,0

1,1,0,0,X,0,0,0,,0,0,1,0,0,0,0,0

1,1,0,1,1,0,1,1,,0,0,1,0,0,0,0,0

X,1,1,0,1,X,1,0,,0,1,0,0,0,0,0,0

X,0,0,1,1,X,0,0,,0,1,0,0,0,0,0,0

1,0,1,0,X,0,0,0,,0,1,0,0,0,0,0,0

X,1,0,X,1,1,0,1,,0,1,0,0,0,0,0,0

0,0,0,1,X,X,0,X,,0,1,0,0,0,0,0,0

0,0,0,0,0,1,X,X,,0,1,0,0,0,0,0,0

1,1,1,1,X,0,1,X,,0,1,0,0,0,0,0,0

1,0,0,0,0,X,X,1,,0,1,0,0,0,0,0,0

0,0,X,1,0,0,X,1,,0,1,0,0,0,0,0,0

X,0,1,0,X,0,1,1,,0,1,0,0,0,0,0,0

1,0,X,X,0,1,0,1,,0,1,0,0,0,0,0,0

0,X,X,0,0,0,0,1,,0,1,0,0,0,0,0,0

X,1,0,1,1,0,X,0,,0,1,0,0,0,0,0,0

1,X,0,1,X,0,0,0,,0,1,0,0,0,0,0,0

0,1,0,1,0,0,1,X,,0,1,0,0,0,0,0,0

1,0,X,0,1,1,1,1,,0,1,0,0,0,0,0,0

X,1,0,0,1,0,1,1,,0,1,0,0,0,0,0,0

1,1,0,1,0,1,X,0,,0,1,0,0,0,0,0,0

0,0,0,X,1,1,1,0,,0,1,0,0,0,0,0,0

1,0,X,1,0,0,1,0,,0,1,0,0,0,0,0,0

X,0,0,0,0,0,1,0,,0,1,0,0,0,0,0,0

X,1,1,0,0,1,0,0,,0,1,0,0,0,0,0,0

X,X,1,1,0,X,1,1,,0,0,1,0,0,0,0,0

X,1,1,1,0,1,1,X,,0,0,1,0,0,0,0,0

0,1,0,0,X,1,X,1,,0,0,1,0,0,0,0,0

X,1,0,X,1,1,0,1,,0,0,1,0,0,0,0,0

1,1,0,X,0,X,0,0,,0,0,1,0,0,0,0,0

1,0,0,0,1,0,X,1,,0,0,1,0,0,0,0,0

0,1,0,X,1,0,1,0,,0,0,1,0,0,0,0,0

1,0,X,0,1,1,0,0,,0,0,1,0,0,0,0,0

0,0,X,1,1,0,0,0,,0,0,1,0,0,0,0,0

0,X,X,1,0,1,1,X,,0,0,1,0,0,0,0,0

X,0,X,1,X,1,1,1,,0,0,1,0,0,0,0,0

X,0,1,1,1,1,1,X,,0,0,1,0,0,0,0,0

1,0,0,0,X,0,1,X,,0,0,1,0,0,0,0,0

1,1,0,1,0,X,0,X,,0,0,1,0,0,0,0,0

0,0,X,0,0,1,0,X,,0,0,1,0,0,0,0,0

1,0,X,1,1,1,X,1,,0,0,1,0,0,0,0,0

1,0,1,X,0,1,X,1,,0,0,1,0,0,0,0,0

1,0,X,0,1,X,1,1,,0,0,1,0,0,0,0,0

0,X,1,1,X,0,1,1,,0,0,1,0,0,0,0,0

1,0,X,X,0,0,1,1,,0,0,1,0,0,0,0,0

0,1,X,1,X,0,0,1,,0,0,1,0,0,0,0,0

X,1,0,X,0,0,0,1,,0,0,1,0,0,0,0,0

0,X,X,1,0,0,0,1,,0,0,1,0,0,0,0,0

1,0,0,1,X,0,X,0,,0,0,1,0,0,0,0,0

0,1,X,1,0,0,X,0,,0,0,1,0,0,0,0,0

1,1,X,0,0,0,X,0,,0,0,1,0,0,0,0,0

0,X,0,X,0,1,1,0,,0,0,1,0,0,0,0,0

1,0,X,X,1,0,1,0,,0,0,1,0,0,0,0,0

0,1,1,0,1,1,0,X,,0,0,1,0,0,0,0,0

0,0,0,1,1,0,X,1,,0,0,1,0,0,0,0,0

1,1,1,0,1,0,X,1,,0,0,1,0,0,0,0,0

0,0,1,0,1,0,X,1,,0,0,1,0,0,0,0,0

0,1,X,0,0,0,1,1,,0,0,1,0,0,0,0,0

1,X,1,1,1,0,0,1,,0,0,1,0,0,0,0,0

0,X,1,0,1,0,0,1,,0,0,1,0,0,0,0,0

1,1,0,X,1,1,1,0,,0,0,1,0,0,0,0,0

1,X,1,0,1,1,1,0,,0,0,1,0,0,0,0,0

X,1,1,1,1,0,1,0,,0,0,1,0,0,0,0,0

0,X,0,1,1,1,0,0,,0,0,1,0,0,0,0,0

1,0,1,X,0,0,0,0,,0,0,1,0,0,0,0,0

X,1,0,0,0,1,1,X,,0,0,0,1,0,0,0,0

1,X,0,1,1,0,1,X,,0,0,0,1,0,0,0,0

0,1,1,X,1,1,0,X,,0,0,0,1,0,0,0,0

X,1,0,1,0,1,0,X,,0,0,0,1,0,0,0,0

X,0,0,1,1,X,1,1,,0,0,0,1,0,0,0,0

1,0,X,0,1,X,1,1,,0,0,0,1,0,0,0,0

1,0,X,1,0,X,1,1,,0,0,0,1,0,0,0,0

0,1,0,X,0,0,1,1,,0,0,0,1,0,0,0,0

0,0,X,1,0,0,0,1,,0,0,0,1,0,0,0,0

1,1,0,1,0,X,0,0,,0,0,0,1,0,0,0,0

1,X,0,1,0,1,X,X,,0,0,0,1,0,0,0,0

1,0,0,X,X,1,0,X,,0,0,0,1,0,0,0,0

X,0,1,X,0,0,0,X,,0,0,0,1,0,0,0,0

X,0,1,1,1,X,X,0,,0,0,0,1,0,0,0,0

0,1,1,X,1,0,1,X,,0,0,0,1,0,0,0,0

1,0,X,1,1,0,1,X,,0,0,0,1,0,0,0,0

0,0,0,0,1,X,0,X,,0,0,0,1,0,0,0,0

0,0,1,0,0,X,0,X,,0,0,0,1,0,0,0,0

X,0,0,0,1,0,X,1,,0,0,0,1,0,0,0,0

0,1,1,X,X,1,1,1,,0,0,0,1,0,0,0,0

X,1,1,0,X,0,1,1,,0,0,0,1,0,0,0,0

1,X,X,0,1,0,1,1,,0,0,0,1,0,0,0,0

0,X,1,0,1,X,0,1,,0,0,0,1,0,0,0,0

0,X,1,1,X,0,0,1,,0,0,0,1,0,0,0,0

1,X,1,X,0,0,0,1,,0,0,0,1,0,0,0,0

1,0,0,0,0,X,X,0,,0,0,0,1,0,0,0,0

1,1,0,0,X,1,X,0,,0,0,0,1,0,0,0,0

1,0,X,1,1,1,X,0,,0,0,0,1,0,0,0,0

0,X,1,1,0,1,X,0,,0,0,0,1,0,0,0,0

1,1,X,0,0,1,X,0,,0,0,0,1,0,0,0,0

0,0,1,X,1,X,1,0,,0,0,0,1,0,0,0,0

0,0,X,1,X,0,1,0,,0,0,0,1,0,0,0,0

X,1,X,1,1,1,0,0,,0,0,0,1,0,0,0,0

0,1,0,0,0,0,0,X,,0,0,0,1,0,0,0,0

1,1,1,0,X,1,0,1,,0,0,0,1,0,0,0,0

0,1,1,0,0,0,X,0,,0,0,0,1,0,0,0,0

0,0,X,0,1,1,1,0,,0,0,0,1,0,0,0,0

0,0,0,X,0,0,1,0,,0,0,0,1,0,0,0,0

X,1,X,1,0,X,1,1,,0,0,0,0,1,0,0,0

X,1,1,0,1,0,X,1,,0,0,0,0,1,0,0,0

0,0,0,1,X,X,0,0,,0,0,0,0,1,0,0,0

1,X,X,0,0,0,0,0,,0,0,0,0,1,0,0,0

0,1,1,1,1,0,X,0,,0,0,0,0,1,0,0,0

0,X,0,0,0,1,1,0,,0,0,0,0,1,0,0,0

0,0,X,X,0,1,1,X,,0,0,0,0,1,0,0,0

X,X,0,1,X,0,1,1,,0,0,0,0,1,0,0,0

X,0,0,X,0,X,0,0,,0,0,0,0,1,0,0,0

1,1,0,0,0,0,X,X,,0,0,0,0,1,0,0,0

0,1,0,X,1,1,0,X,,0,0,0,0,1,0,0,0

1,1,0,1,0,X,X,1,,0,0,0,0,1,0,0,0

1,0,1,1,X,0,X,1,,0,0,0,0,1,0,0,0

X,1,1,0,1,X,1,1,,0,0,0,0,1,0,0,0

1,1,X,X,1,0,1,1,,0,0,0,0,1,0,0,0

0,X,1,0,X,0,0,1,,0,0,0,0,1,0,0,0

X,0,X,0,0,1,1,0,,0,0,0,0,1,0,0,0

1,1,X,1,X,1,0,0,,0,0,0,0,1,0,0,0

X,X,1,0,1,1,0,0,,0,0,0,0,1,0,0,0

1,X,0,0,X,0,0,0,,0,0,0,0,1,0,0,0

X,1,1,1,1,1,0,1,,0,0,0,0,1,0,0,0

X,0,0,0,1,0,0,1,,0,0,0,0,1,0,0,0

0,0,1,X,1,1,0,0,,0,0,0,0,1,0,0,0

X,0,0,1,1,0,1,X,,0,0,0,0,0,1,0,0

1,1,0,0,X,0,X,1,,0,0,0,0,0,1,0,0

0,0,X,X,1,0,0,0,,0,0,0,1,0,0,0,0

0,0,0,0,0,1,X,1,,0,0,0,1,0,0,0,0

1,1,1,1,X,1,X,0,,0,0,0,0,0,1,0,0

0,0,0,1,0,0,X,1,,0,0,0,1,0,0,0,0

1,1,1,X,1,1,1,0,,0,0,0,1,0,0,0,0

X,0,1,0,0,1,1,0,,0,0,0,1,0,0,0,0

X,1,0,0,1,0,0,0,,0,0,0,1,0,0,0,0

0,1,1,0,X,1,0,X,,0,0,0,0,1,0,0,0

1,1,0,1,X,X,1,1,,0,0,0,0,1,0,0,0

0,X,0,X,0,0,0,0,,0,0,0,0,1,0,0,0

1,0,1,1,1,0,0,X,,0,0,0,0,1,0,0,0

1,X,0,0,1,1,1,0,,0,0,0,0,1,0,0,0

0,1,1,0,X,0,1,0,,0,0,0,0,1,0,0,0

1,0,0,0,X,X,0,X,,0,0,0,0,1,0,0,0

X,0,0,X,0,0,X,0,,0,0,0,0,1,0,0,0

0,1,0,0,1,0,X,X,,0,0,0,0,1,0,0,0

X,0,0,0,1,1,1,X,,0,0,0,0,1,0,0,0

0,0,1,0,X,0,0,X,,0,0,0,0,1,0,0,0

0,0,0,1,X,1,X,1,,0,0,0,0,1,0,0,0

X,0,1,1,1,0,X,1,,0,0,0,0,1,0,0,0

1,1,X,X,0,1,1,1,,0,0,0,0,1,0,0,0

1,0,1,X,X,1,0,1,,0,0,0,0,1,0,0,0

X,0,1,1,0,X,1,0,,0,0,0,0,1,0,0,0

1,X,0,X,0,0,1,0,,0,0,0,0,1,0,0,0

1,X,0,1,X,1,0,0,,0,0,0,0,1,0,0,0

1,X,X,1,0,1,0,0,,0,0,0,0,1,0,0,0

0,0,1,0,0,X,1,1,,0,0,0,0,1,0,0,0

1,X,1,0,0,1,0,1,,0,0,0,0,1,0,0,0

0,X,1,1,1,1,1,0,,0,0,0,0,1,0,0,0

1,X,X,1,X,1,0,0,,0,0,0,0,0,1,0,0

0,0,1,0,0,X,0,X,,0,0,0,0,0,1,0,0

1,1,1,1,X,1,X,0,,0,0,0,0,0,1,0,0

X,1,1,0,0,0,X,0,,0,0,0,0,0,1,0,0

0,X,1,1,1,1,0,1,,0,0,0,0,0,1,0,0

X,0,0,0,0,0,1,0,,0,0,0,0,0,1,0,0

0,1,0,0,1,1,X,X,,0,0,0,0,0,1,0,0

0,1,1,1,1,X,0,X,,0,0,0,0,0,1,0,0

X,1,1,1,0,1,0,X,,0,0,0,0,0,1,0,0

0,X,X,1,1,0,1,1,,0,0,0,0,0,1,0,0

0,X,0,X,0,1,0,1,,0,0,0,0,0,1,0,0

1,X,0,0,X,0,0,1,,0,0,0,0,0,1,0,0

0,0,1,X,1,0,X,0,,0,0,0,0,0,1,0,0

X,0,1,0,1,0,X,0,,0,0,0,0,0,1,0,0

0,X,0,X,1,1,0,0,,0,0,0,0,0,1,0,0

0,1,X,0,X,0,0,0,,0,0,0,0,0,1,0,0

1,0,1,1,1,1,1,X,,0,0,0,0,0,1,0,0

1,1,1,0,1,1,0,X,,0,0,0,0,0,1,0,0

1,0,0,0,X,1,1,1,,0,0,0,0,0,1,0,0

0,0,X,1,0,1,1,1,,0,0,0,0,0,1,0,0

1,0,1,0,X,0,1,1,,0,0,0,0,0,1,0,0

1,0,0,1,0,X,0,1,,0,0,0,0,0,1,0,0

1,1,0,0,X,1,1,0,,0,0,0,0,0,1,0,0

0,X,0,1,0,0,0,0,,0,0,0,0,0,1,0,0

X,0,1,1,0,0,0,X,,0,0,0,0,0,0,1,0

X,0,1,0,1,X,0,1,,0,0,0,0,0,0,1,0

0,X,0,X,1,0,1,0,,0,0,0,0,0,0,1,0

0,1,X,X,0,1,0,0,,0,0,0,0,0,0,1,0

0,0,0,1,1,1,0,X,,0,0,0,0,0,0,1,0

1,0,0,X,0,1,0,1,,0,0,0,0,0,0,1,0

1,0,1,0,1,X,1,0,,0,0,0,0,0,0,1,0

1,0,0,1,X,0,0,0,,0,0,0,0,0,0,1,0

0,1,0,X,0,X,1,X,,0,0,0,0,0,0,1,0

0,1,X,0,X,X,1,1,,0,0,0,0,0,0,1,0

0,0,0,X,0,X,1,0,,0,0,0,0,0,0,1,0

0,1,1,0,0,X,1,0,,0,0,0,0,0,0,1,0

1,0,X,X,1,X,0,0,,0,0,0,0,0,0,1,0

0,0,0,X,0,1,X,0,,0,0,0,0,0,0,1,0

0,1,0,1,0,X,0,X,,0,0,0,0,0,0,1,0

X,X,1,0,1,1,1,1,,0,0,0,0,0,0,1,0

0,X,X,0,0,0,1,1,,0,0,0,0,0,0,1,0

1,X,1,1,X,0,0,1,,0,0,0,0,0,0,1,0

X,1,0,X,1,0,0,1,,0,0,0,0,0,0,1,0

1,1,X,1,1,0,X,0,,0,0,0,0,0,0,1,0

0,X,0,1,1,X,1,0,,0,0,0,0,0,0,1,0

1,X,0,X,0,1,0,0,,0,0,0,0,0,0,1,0

X,1,0,X,0,0,0,0,,0,0,0,0,0,0,1,0

1,0,1,1,0,0,1,X,,0,0,0,0,0,0,1,0

0,0,1,0,X,1,1,1,,0,0,0,0,0,0,1,0

0,1,1,X,1,1,1,1,,0,0,0,0,0,0,1,0

1,1,X,0,0,1,1,1,,0,0,0,0,0,0,1,0

1,1,X,1,0,0,1,1,,0,0,0,0,0,0,1,0

0,0,X,0,1,0,0,1,,0,0,0,0,0,0,1,0

0,X,1,1,0,0,1,0,,0,0,0,0,0,0,1,0

0,1,0,X,X,0,1,X,,0,0,0,0,0,0,0,1

X,1,0,0,1,X,1,1,,0,0,0,0,0,0,0,1

1,1,1,X,X,1,1,0,,0,0,0,0,0,0,0,1

0,X,0,0,X,1,0,0,,0,0,0,0,0,0,0,1

1,1,1,1,1,0,1,X,,0,0,0,0,0,0,0,1

1,0,0,1,X,0,1,1,,0,0,0,0,0,0,0,1

1,1,X,0,0,0,0,1,,0,0,0,0,0,0,0,1

1,0,X,1,0,1,1,0,,0,0,0,0,0,0,0,1

0,1,X,1,0,1,X,X,,0,0,0,0,0,0,0,1

0,X,0,X,0,0,X,1,,0,0,0,0,0,0,0,1

0,X,1,0,1,1,1,X,,0,0,0,0,0,0,0,1

0,0,X,0,0,0,1,X,,0,0,0,0,0,0,1,0
0,1,X,0,1,1,X,1,,0,0,0,0,0,0,1,0
1,X,0,0,0,1,X,1,,0,0,0,0,0,0,1,0
0,X,X,1,0,1,1,1,,0,0,0,0,0,0,1,0
0,1,1,1,X,X,0,1,,0,0,0,0,0,0,1,0
0,1,1,X,1,0,X,0,,0,0,0,0,0,0,1,0
X,1,0,1,1,X,1,0,,0,0,0,0,0,0,1,0
X,0,1,1,X,1,1,0,,0,0,0,0,0,0,1,0
0,1,X,1,X,0,0,0,,0,0,0,0,0,0,1,0
1,0,0,0,1,1,0,X,,0,0,0,0,0,0,1,0
1,1,0,1,X,0,0,1,,0,0,0,0,0,0,1,0
1,0,0,0,0,0,X,0,,0,0,0,0,0,0,1,0
1,1,0,0,X,0,0,0,,0,0,0,0,0,0,1,0
0,X,1,X,1,0,X,1,,0,0,0,0,0,0,0,1
1,X,0,X,1,X,1,0,,0,0,0,0,0,0,0,1
1,1,1,0,1,X,1,X,,0,0,0,0,0,0,0,1
X,0,1,0,0,0,1,X,,0,0,0,0,0,0,0,1
1,0,X,0,1,1,0,X,,0,0,0,0,0,0,0,1
X,0,1,0,0,0,X,1,,0,0,0,0,0,0,0,1
0,1,1,X,1,X,0,1,,0,0,0,0,0,0,0,1
1,X,X,1,0,0,0,0,,0,0,0,0,0,0,0,1
1,1,X,1,0,0,1,1,,0,0,0,0,0,0,0,1
1,1,1,X,0,1,1,0,,0,0,0,0,0,0,0,1
0,0,X,1,1,X,1,X,,0,0,0,0,0,0,0,1
0,1,X,1,1,X,X,1,,0,0,0,0,0,0,0,1
X,0,1,1,1,0,1,X,,0,0,0,0,0,0,0,1
0,1,1,1,X,1,X,1,,0,0,0,0,0,0,0,1
0,0,0,X,0,1,X,1,,0,0,0,0,0,0,0,1
X,0,0,X,1,1,1,1,,0,0,0,0,0,0,0,1
0,0,0,X,0,X,0,1,,0,0,0,0,0,0,0,1
0,1,X,1,X,0,0,1,,0,0,0,0,0,0,0,1

1,0,1,X,1,1,X,1,,0,0,0,0,0,0,1,0
0,0,1,X,0,1,X,1,,0,0,0,0,0,0,1,0
1,X,0,0,X,1,1,1,,0,0,0,0,0,0,1,0
X,1,X,1,0,1,1,1,,0,0,0,0,0,0,1,0
1,0,1,1,0,X,X,0,,0,0,0,0,0,0,1,0
X,1,0,1,0,0,X,0,,0,0,0,0,0,0,1,0
0,0,1,X,0,X,1,0,,0,0,0,0,0,0,1,0
X,1,X,1,0,1,0,0,,0,0,0,0,0,0,1,0
1,1,1,0,1,1,0,X,,0,0,0,0,0,0,1,0
1,1,X,1,1,1,0,1,,0,0,0,0,0,0,1,0
0,0,0,0,X,0,0,1,,0,0,0,0,0,0,1,0
0,0,1,0,X,0,0,0,,0,0,0,0,0,0,1,0
1,1,X,0,1,0,0,0,,0,0,0,0,0,0,1,0
1,0,0,1,X,X,X,0,,0,0,0,0,0,0,0,1
X,0,X,0,1,X,0,0,,0,0,0,0,0,0,0,1
0,1,0,1,X,1,1,X,,0,0,0,0,0,0,0,1
1,1,0,0,0,X,0,X,,0,0,0,0,0,0,0,1
0,0,X,1,0,1,0,X,,0,0,0,0,0,0,0,1
0,X,0,X,1,1,1,1,,0,0,0,0,0,0,0,1
X,1,1,1,1,X,0,1,,0,0,0,0,0,0,0,1
1,1,0,0,0,1,X,1,,0,0,0,0,0,0,0,1
1,1,X,0,0,1,0,1,,0,0,0,0,0,0,0,1
0,X,0,0,1,1,0,0,,0,0,0,0,0,0,0,1
0,0,X,X,1,0,1,X,,0,0,0,0,0,0,0,1
X,0,0,1,X,0,X,0,,0,0,0,0,0,0,0,1
X,0,0,0,1,0,0,X,,0,0,0,0,0,0,0,1
0,X,0,1,1,1,X,1,,0,0,0,0,0,0,0,1
1,1,X,0,1,0,X,1,,0,0,0,0,0,0,0,1
0,0,X,1,X,0,1,1,,0,0,0,0,0,0,0,1
X,0,X,1,0,1,0,1,,0,0,0,0,0,0,0,1
X,1,X,1,1,0,0,1,,0,0,0,0,0,0,0,1

0,X,1,0,1,X,1,0,,0,0,0,0,0,0,0,1

0,X,0,0,X,0,1,0,,0,0,0,0,0,0,0,1

0,0,0,X,X,1,0,0,,0,0,0,0,0,0,0,1

1,X,0,0,X,0,0,0,,0,0,0,0,0,0,0,1

0,1,1,0,0,1,1,X,,0,0,0,0,0,0,0,1

1,0,1,1,0,X,1,1,,0,0,0,0,0,0,0,1

0,1,X,0,0,0,1,1,,0,0,0,0,0,0,0,1

1,X,1,0,1,1,0,0,,0,0,0,0,0,0,0,1

1,1,1,1,X,0,0,0,,0,0,0,0,0,0,0,1

A.2 Мінімізована таблиця істинності другої підстановки

x7,x6,x5,x4,x3,x2,x1,x0,,s7,s6,s5,s4,s3,s2,s1,s0

X,1,1,1,1,1,0,X,,1,0,0,0,0,0,0,0

1,1,X,1,1,1,X,1,,1,0,0,0,0,0,0,0

0,0,X,1,X,0,1,1,,1,0,0,0,0,0,0,0

X,1,1,1,0,X,0,1,,1,0,0,0,0,0,0,0

0,0,X,0,0,X,0,1,,1,0,0,0,0,0,0,0

1,1,0,0,0,X,X,0,,1,0,0,0,0,0,0,0

1,1,0,0,X,1,X,0,,1,0,0,0,0,0,0,0

1,X,1,0,1,0,X,0,,1,0,0,0,0,0,0,0

1,1,X,X,1,0,1,0,,1,0,0,0,0,0,0,0

0,0,0,X,1,X,0,0,,1,0,0,0,0,0,0,0

1,X,1,1,0,1,1,0,,1,0,0,0,0,0,0,0

0,1,X,1,0,0,1,0,,1,0,0,0,0,0,0,0

0,1,1,0,0,X,0,0,,1,0,0,0,0,0,0,0

0,1,0,X,0,1,0,0,,1,0,0,0,0,0,0,0

0,0,0,1,0,1,1,0,,1,0,0,0,0,0,0,0

1,0,0,1,0,1,0,0,,1,0,0,0,0,0,0,0

1,1,0,X,1,X,1,X,,1,0,0,0,0,0,0,0

0,0,1,1,1,X,1,X,,1,0,0,0,0,0,0,0

1,X,0,1,1,1,X,1,,1,0,0,0,0,0,0,0

0,X,0,0,1,1,X,1,,1,0,0,0,0,0,0,0

1,1,0,X,0,1,X,1,,1,0,0,0,0,0,0,0

1,1,X,0,0,1,X,1,,1,0,0,0,0,0,0,0

0,X,0,1,1,X,1,1,,1,0,0,0,0,0,0,0

0,X,0,0,0,X,1,1,,1,0,0,0,0,0,0,0

X,0,X,0,0,1,1,1,,1,0,0,0,0,0,0,0

X,0,X,0,1,0,1,1,,1,0,0,0,0,0,0,0

X,0,0,0,0,X,0,1,,1,0,0,0,0,0,0,0

X,1,1,X,1,1,0,1,,1,0,0,0,0,0,0,0

1,0,0,X,X,0,0,1,,1,0,0,0,0,0,0,0

0,1,1,X,1,1,X,0,,1,0,0,0,0,0,0,0

1,X,0,1,1,X,1,0,,1,0,0,0,0,0,0,0

0,1,X,0,X,1,1,0,,1,0,0,0,0,0,0,0

X,1,X,0,1,1,1,0,,1,0,0,0,0,0,0,0

X,0,0,0,X,0,1,0,,1,0,0,0,0,0,0,0

0,0,0,0,X,X,0,0,,1,0,0,0,0,0,0,0

X,0,1,0,X,1,0,0,,1,0,0,0,0,0,0,0

X,X,1,0,0,1,0,0,,1,0,0,0,0,0,0,0

0,X,X,1,1,0,0,0,,1,0,0,0,0,0,0,0

1,1,1,0,0,0,1,X,,1,0,0,0,0,0,0,0

0,1,0,0,1,0,0,X,,1,0,0,0,0,0,0,0

1,0,1,0,1,1,X,1,,1,0,0,0,0,0,0,0

1,0,1,1,0,1,X,1,,1,0,0,0,0,0,0,0

1,0,0,1,0,X,1,1,,1,0,0,0,0,0,0,0

0,1,X,1,0,1,1,1,,1,0,0,0,0,0,0,0

0,1,1,0,X,0,1,1,,1,0,0,0,0,0,0,0

1,0,1,1,0,0,X,0,,1,0,0,0,0,0,0,0

1,X,1,1,0,0,0,0,,1,0,0,0,0,0,0,0

X,X,0,1,0,X,0,0,,0,1,0,0,0,0,0,0

1,1,1,1,X,1,0,X,,0,1,0,0,0,0,0,0

1,0,X,X,1,0,1,1,,0,1,0,0,0,0,0,0

1,X,1,1,0,X,0,1,,0,1,0,0,0,0,0,0

1,0,X,0,0,1,X,0,,0,1,0,0,0,0,0,0

X,1,1,0,0,0,X,0,,0,1,0,0,0,0,0,0

0,1,1,1,1,0,0,X,,0,1,0,0,0,0,0,0

0,0,1,1,1,X,1,0,,0,1,0,0,0,0,0,0

1,0,1,X,1,0,0,0,,0,1,0,0,0,0,0,0

X,X,1,0,0,0,1,X,,0,1,0,0,0,0,0,0

1,0,1,0,1,0,X,X,,0,1,0,0,0,0,0,0

1,1,1,1,0,X,X,1,,0,1,0,0,0,0,0,0

X,0,1,0,1,1,X,1,,0,1,0,0,0,0,0,0

0,X,0,0,X,1,1,1,,0,1,0,0,0,0,0,0

X,1,1,1,X,0,0,1,,0,1,0,0,0,0,0,0

1,1,0,0,1,X,X,0,,0,1,0,0,0,0,0,0

0,X,0,1,1,1,X,0,,0,1,0,0,0,0,0,0

X,1,0,X,1,0,1,0,,0,1,0,0,0,0,0,0

0,0,X,0,1,X,0,0,,0,1,0,0,0,0,0,0

0,0,0,0,0,1,0,X,,0,1,0,0,0,0,0,0

0,1,X,1,1,1,1,1,,0,1,0,0,0,0,0,0

1,1,X,0,1,1,0,1,,0,1,0,0,0,0,0,0

1,0,X,1,0,1,0,1,,0,1,0,0,0,0,0,0

1,0,1,0,X,1,X,X,,0,0,1,0,0,0,0,0

1,1,0,1,0,X,1,X,,0,0,1,0,0,0,0,0

0,1,X,1,1,1,0,X,,0,0,1,0,0,0,0,0

0,0,1,1,X,X,1,1,,0,0,1,0,0,0,0,0

1,1,0,X,X,0,0,1,,0,0,1,0,0,0,0,0

1,1,1,X,0,0,1,1,,1,0,0,0,0,0,0,0

X,0,1,1,1,0,0,0,,1,0,0,0,0,0,0,0

0,1,X,0,X,0,1,X,,0,1,0,0,0,0,0,0

0,0,0,1,1,X,0,X,,0,1,0,0,0,0,0,0

1,X,0,1,X,0,1,1,,0,1,0,0,0,0,0,0

0,X,1,X,0,0,1,1,,0,1,0,0,0,0,0,0

1,X,0,0,X,0,0,1,,0,1,0,0,0,0,0,0

0,0,0,X,0,0,X,0,,0,1,0,0,0,0,0,0

X,1,X,1,0,1,0,0,,0,1,0,0,0,0,0,0

0,0,0,X,0,1,1,1,,0,1,0,0,0,0,0,0

0,0,1,1,X,1,1,0,,0,1,0,0,0,0,0,0

0,0,X,1,0,0,0,0,,0,1,0,0,0,0,0,0

X,1,1,0,X,X,1,0,,0,1,0,0,0,0,0,0

1,X,1,0,1,1,1,X,,0,1,0,0,0,0,0,0

1,1,1,1,X,1,X,1,,0,1,0,0,0,0,0,0

0,1,0,1,X,0,X,1,,0,1,0,0,0,0,0,0

X,0,X,0,1,1,1,1,,0,1,0,0,0,0,0,0

X,1,0,X,0,0,0,1,,0,1,0,0,0,0,0,0

0,0,1,X,1,1,X,0,,0,1,0,0,0,0,0,0

0,X,0,0,1,0,X,0,,0,1,0,0,0,0,0,0

X,0,X,0,0,0,1,0,,0,1,0,0,0,0,0,0

1,1,X,0,X,0,0,0,,0,1,0,0,0,0,0,0

0,1,1,0,0,1,X,1,,0,1,0,0,0,0,0,0

0,0,1,0,0,X,0,1,,0,1,0,0,0,0,0,0

0,0,1,X,0,1,0,1,,0,1,0,0,0,0,0,0

1,1,0,X,0,1,1,0,,0,1,0,0,0,0,0,0

1,0,1,X,0,1,X,X,,0,0,1,0,0,0,0,0

1,1,0,1,X,1,1,X,,0,0,1,0,0,0,0,0

1,1,X,1,1,0,0,X,,0,0,1,0,0,0,0,0

X,0,1,0,X,1,0,1,,0,0,1,0,0,0,0,0

1,0,X,1,1,1,0,1,,0,0,1,0,0,0,0,0

1,0,0,X,0,0,0,0,,0,0,1,0,0,0,0,0

0,1,1,0,1,X,1,X,,0,0,1,0,0,0,0,0

X,1,0,0,0,1,1,X,,0,0,1,0,0,0,0,0

1,1,X,0,0,0,0,X,,0,0,1,0,0,0,0,0

X,1,0,0,0,X,1,1,,0,0,1,0,0,0,0,0

1,X,X,0,0,1,1,1,,0,0,1,0,0,0,0,0

0,0,X,X,1,0,1,1,,0,0,1,0,0,0,0,0

1,X,1,0,X,1,0,1,,0,0,1,0,0,0,0,0

X,X,1,0,0,0,0,1,,0,0,1,0,0,0,0,0

0,X,0,1,1,X,1,0,,0,0,1,0,0,0,0,0

1,X,1,1,X,1,1,0,,0,0,1,0,0,0,0,0

1,X,0,X,1,1,1,0,,0,0,1,0,0,0,0,0

1,1,1,X,X,0,0,0,,0,0,1,0,0,0,0,0

0,X,0,0,1,1,1,1,,0,0,1,0,0,0,0,0

0,1,1,X,0,0,1,1,,0,0,1,0,0,0,0,0

X,1,1,1,1,0,0,1,,0,0,1,0,0,0,0,0

0,X,0,0,0,0,1,0,,0,0,1,0,0,0,0,0

1,1,X,0,1,1,0,0,,0,0,1,0,0,0,0,0

0,0,X,0,1,0,0,0,,0,0,1,0,0,0,0,0

1,1,1,0,1,1,X,X,,0,0,0,1,0,0,0,0

0,0,X,0,X,0,1,1,,0,0,0,1,0,0,0,0

1,0,1,1,X,0,1,1,,0,0,0,1,0,0,0,0

0,1,1,0,1,0,X,0,,0,0,0,0,1,0,0,0,0

1,0,0,1,0,0,X,X,,0,0,0,1,0,0,0,0

1,0,0,X,1,1,1,X,,0,0,0,1,0,0,0,0

0,X,0,1,1,0,1,X,,0,0,0,1,0,0,0,0

X,0,1,0,0,0,0,X,,0,0,0,1,0,0,0,0

X,X,0,1,0,0,1,1,,0,0,0,1,0,0,0,0

0,0,X,1,1,X,0,1,,0,0,0,1,0,0,0,0

0,1,0,X,X,1,0,1,,0,0,0,1,0,0,0,0

X,0,0,0,X,0,0,1,,0,0,0,1,0,0,0,0

0,0,1,1,0,0,X,X,,0,0,1,0,0,0,0,0

X,0,1,0,0,1,1,X,,0,0,1,0,0,0,0,0

1,0,X,0,0,1,0,X,,0,0,1,0,0,0,0,0

1,1,X,1,0,X,1,1,,0,0,1,0,0,0,0,0

X,1,0,X,0,1,1,1,,0,0,1,0,0,0,0,0

X,0,0,1,X,0,1,1,,0,0,1,0,0,0,0,0

0,1,0,X,0,X,0,1,,0,0,1,0,0,0,0,0

X,1,X,1,0,1,0,1,,0,0,1,0,0,0,0,0

1,0,1,X,1,0,X,0,,0,0,1,0,0,0,0,0

X,0,1,0,0,X,1,0,,0,0,1,0,0,0,0,0

0,X,1,0,X,1,1,0,,0,0,1,0,0,0,0,0

0,1,X,X,1,0,1,0,,0,0,1,0,0,0,0,0

0,X,1,X,0,0,0,0,,0,0,1,0,0,0,0,0

1,X,1,1,1,0,1,1,,0,0,1,0,0,0,0,0

0,0,0,0,X,0,0,1,,0,0,1,0,0,0,0,0

0,1,1,1,0,X,1,0,,0,0,1,0,0,0,0,0

0,0,0,1,X,1,0,0,,0,0,1,0,0,0,0,0

0,X,0,0,0,1,0,0,,0,0,1,0,0,0,0,0

1,1,0,1,X,X,0,X,,0,0,0,1,0,0,0,0

1,0,0,0,X,1,1,X,,0,0,0,1,0,0,0,0

X,0,0,0,X,1,1,0,,0,0,0,1,0,0,0,0

1,1,X,1,1,0,0,1,,0,0,0,1,0,0,0,0

0,1,X,1,0,1,1,0,,0,0,0,1,0,0,0,0

0,0,0,0,1,X,1,X,,0,0,0,1,0,0,0,0

1,X,1,0,1,1,1,X,,0,0,0,1,0,0,0,0

0,0,0,X,1,1,0,X,,0,0,0,1,0,0,0,0

0,X,1,1,X,1,1,1,,0,0,0,1,0,0,0,0

0,0,0,1,X,X,0,1,,0,0,0,1,0,0,0,0

X,1,0,0,1,X,0,1,,0,0,0,1,0,0,0,0

X,X,0,1,1,1,0,1,,0,0,0,1,0,0,0,0

0,0,1,X,1,1,X,0,,0,0,0,1,0,0,0,0

1,1,X,0,0,0,X,0,,0,0,0,1,0,0,0,0

X,X,0,0,1,1,1,0,,0,0,0,1,0,0,0,0

1,X,0,X,1,0,1,0,,0,0,0,1,0,0,0,0

1,1,X,1,0,X,0,0,,0,0,0,1,0,0,0,0

0,X,1,X,1,0,0,0,,0,0,0,1,0,0,0,0

0,0,1,1,0,X,1,1,,0,0,0,1,0,0,0,0

X,1,1,1,1,1,1,1,,0,0,0,1,0,0,0,0

X,1,1,0,0,0,1,1,,0,0,0,1,0,0,0,0

0,0,1,X,0,1,0,1,,0,0,0,1,0,0,0,0

0,1,1,1,0,0,X,0,,0,0,0,1,0,0,0,0

X,1,1,0,0,1,1,0,,0,0,0,1,0,0,0,0

0,X,1,0,0,1,0,0,,0,0,0,1,0,0,0,0

0,1,0,X,0,X,X,0,,0,0,0,0,1,0,0,0

1,0,0,X,1,0,1,X,,0,0,0,0,1,0,0,0

0,0,0,0,X,0,0,X,,0,0,0,0,1,0,0,0

0,X,X,1,1,1,1,1,,0,0,0,0,1,0,0,0

1,0,X,0,1,X,1,0,,0,0,0,0,1,0,0,0

1,0,1,X,X,1,0,0,,0,0,0,0,1,0,0,0

1,1,0,1,X,1,1,1,,0,0,0,0,1,0,0,0

1,1,0,X,0,0,0,1,,0,0,0,0,1,0,0,0

1,0,0,0,0,1,1,1,,0,0,0,0,1,0,0,0

1,X,1,0,1,1,1,X,,0,0,0,0,1,0,0,0

0,0,0,0,0,X,0,X,,0,0,0,0,1,0,0,0

0,1,0,X,1,0,X,1,,0,0,0,0,1,0,0,0

0,0,1,X,X,1,1,1,,0,0,0,0,1,0,0,0

0,X,X,0,1,0,0,1,,0,0,0,0,1,0,0,0

X,X,0,0,0,0,0,1,,0,0,0,0,1,0,0,0

X,1,0,0,0,1,X,0,,0,0,0,0,1,0,0,0

X,0,0,0,0,0,X,0,,0,0,0,0,1,0,0,0

1,X,0,1,X,0,1,0,,0,0,0,0,1,0,0,0

X,0,X,0,0,1,0,0,,0,0,0,0,1,0,0,0

1,0,1,0,X,X,1,0,,0,0,0,1,0,0,0,0

X,0,0,1,X,0,1,0,,0,0,0,1,0,0,0,0

0,1,0,0,X,X,0,0,,0,0,0,1,0,0,0,0

0,X,1,1,X,0,0,0,,0,0,0,1,0,0,0,0

1,0,0,0,1,0,0,X,,0,0,0,1,0,0,0,0

0,1,0,0,0,X,1,1,,0,0,0,1,0,0,0,0

1,1,1,X,0,0,1,1,,0,0,0,1,0,0,0,0

0,1,1,1,0,X,0,1,,0,0,0,1,0,0,0,0

1,X,1,0,0,1,0,1,,0,0,0,1,0,0,0,0

1,0,1,X,0,1,1,0,,0,0,0,1,0,0,0,0

1,0,1,1,X,1,0,0,,0,0,0,1,0,0,0,0

1,X,0,0,0,1,0,0,,0,0,0,1,0,0,0,0

0,0,1,1,1,X,1,X,,0,0,0,0,1,0,0,0

0,X,1,0,1,1,0,X,,0,0,0,0,1,0,0,0

0,1,1,X,0,0,X,1,,0,0,0,0,1,0,0,0

0,0,1,1,X,0,X,0,,0,0,0,0,1,0,0,0

X,1,1,0,0,X,1,0,,0,0,0,0,1,0,0,0

1,1,0,0,0,0,X,1,,0,0,0,0,1,0,0,0

1,X,1,1,1,0,0,1,,0,0,0,0,1,0,0,0

1,1,X,0,1,0,0,0,,0,0,0,0,1,0,0,0

X,X,0,1,1,X,1,1,,0,0,0,0,1,0,0,0

0,1,0,1,X,0,1,X,,0,0,0,0,1,0,0,0

1,X,0,1,1,1,X,1,,0,0,0,0,1,0,0,0

X,0,0,1,1,0,X,1,,0,0,0,0,1,0,0,0

0,0,X,X,1,1,1,1,,0,0,0,0,1,0,0,0

X,1,X,0,0,0,0,1,,0,0,0,0,1,0,0,0

0,1,X,1,0,1,X,0,,0,0,0,0,1,0,0,0

1,X,0,1,1,0,X,0,,0,0,0,0,1,0,0,0

1,1,X,1,1,X,1,0,,0,0,0,0,1,0,0,0

0,1,X,X,1,0,1,0,,0,0,0,0,1,0,0,0

1,0,X,X,0,0,0,0,,0,0,0,0,1,0,0,0

1,0,1,1,0,1,1,X,,0,0,0,0,1,0,0,0

1,1,1,1,0,1,0,X,,0,0,0,0,1,0,0,0

1,1,1,0,1,1,X,1,,0,0,0,0,1,0,0,0

1,0,X,1,0,0,1,1,,0,0,0,0,1,0,0,0

0,X,0,1,0,1,0,1,,0,0,0,0,1,0,0,0

X,0,0,1,0,1,1,0,,0,0,0,0,1,0,0,0

0,X,1,1,0,X,1,X,,0,0,0,0,0,1,0,0

X,0,1,1,X,0,X,1,,0,0,0,0,0,1,0,0

1,1,X,0,1,0,0,X,,0,0,0,0,0,1,0,0

1,1,X,0,0,X,0,1,,0,0,0,0,0,1,0,0

1,0,0,1,0,X,0,0,,0,0,0,0,0,1,0,0

1,1,1,1,0,0,0,0,,0,0,0,0,0,1,0,0

0,X,0,1,X,1,0,X,,0,0,0,0,0,1,0,0

X,X,X,1,1,1,0,0,,0,0,0,0,0,1,0,0

1,1,0,1,1,X,1,X,,0,0,0,0,0,1,0,0

0,0,X,1,0,0,1,X,,0,0,0,0,0,1,0,0

0,0,1,1,X,0,0,X,,0,0,0,0,0,1,0,0

1,X,0,0,1,0,X,1,,0,0,0,0,0,1,0,0

0,0,0,1,X,X,0,1,,0,0,0,0,0,1,0,0

0,X,1,0,1,X,0,1,,0,0,0,0,0,1,0,0

1,X,X,1,1,0,0,1,,0,0,0,0,0,1,0,0

1,X,1,0,1,0,X,0,,0,0,0,0,0,1,0,0

X,1,1,X,0,1,1,0,,0,0,0,0,0,1,0,0

0,0,1,X,1,X,0,0,,0,0,0,0,0,1,0,0

X,X,1,0,0,1,0,0,,0,0,0,0,0,1,0,0

1,0,0,1,0,1,X,1,,0,0,0,0,0,1,0,0

1,0,0,X,0,1,1,1,,0,0,0,0,0,1,0,0

X,0,1,0,1,1,0,1,,0,0,0,0,0,1,0,0

0,1,0,1,0,0,X,0,,0,0,0,0,0,1,0,0

1,1,1,X,1,1,0,0,,0,0,0,0,0,1,0,0

X,0,0,0,0,X,0,X,,0,0,0,0,0,0,1,0

1,0,0,0,1,1,0,X,,0,0,0,0,0,1,0,0

0,0,1,1,0,1,0,X,,0,0,0,0,0,1,0,0

0,1,0,0,X,1,1,1,,0,0,0,0,0,1,0,0

X,0,1,0,0,0,1,1,,0,0,0,0,0,1,0,0

1,X,1,0,0,1,0,1,,0,0,0,0,0,1,0,0

X,1,0,1,1,1,0,0,,0,0,0,0,0,1,0,0

0,X,1,1,0,X,X,1,,0,0,0,0,0,1,0,0

1,0,1,1,1,1,X,X,,0,0,0,0,0,1,0,0

0,1,1,X,X,0,1,1,,0,0,0,0,0,1,0,0

0,0,1,X,0,0,1,0,,0,0,0,0,0,1,0,0

0,1,1,0,0,X,0,0,,0,0,0,0,0,1,0,0

0,0,0,0,0,0,0,0,,0,0,0,0,0,1,0,0

X,X,1,1,1,X,1,1,,0,0,0,0,0,1,0,0

0,0,0,1,1,1,X,X,,0,0,0,0,0,1,0,0

1,1,0,1,X,0,1,X,,0,0,0,0,0,1,0,0

0,0,0,1,1,X,0,X,,0,0,0,0,0,1,0,0

1,1,1,X,0,1,X,1,,0,0,0,0,0,1,0,0

1,X,0,1,0,X,1,1,,0,0,0,0,0,1,0,0

0,0,1,0,X,X,0,1,,0,0,0,0,0,1,0,0

0,1,1,X,X,1,0,1,,0,0,0,0,0,1,0,0

1,0,0,0,X,1,X,0,,0,0,0,0,0,1,0,0

X,0,0,0,1,X,1,0,,0,0,0,0,0,1,0,0

X,0,0,X,1,0,1,0,,0,0,0,0,0,1,0,0

0,1,0,X,X,1,0,0,,0,0,0,0,0,1,0,0

0,1,0,0,1,1,X,1,,0,0,0,0,0,1,0,0

0,0,1,X,1,1,1,1,,0,0,0,0,0,1,0,0

X,1,0,0,1,0,1,1,,0,0,0,0,0,1,0,0

0,1,1,1,1,0,X,0,,0,0,0,0,0,1,0,0

1,1,X,0,0,0,1,0,,0,0,0,0,0,1,0,0

1,X,1,X,1,0,1,X,,0,0,0,0,0,0,1,0

0,0,X,0,X,1,X,0,,0,0,0,0,0,0,1,0

1,X,1,1,0,0,0,X,,0,0,0,0,0,0,1,0

0,0,X,0,X,0,0,1,,0,0,0,0,0,0,1,0

0,0,0,X,1,X,1,0,,0,0,0,0,0,0,1,0

0,X,0,1,1,1,0,1,,0,0,0,0,0,0,1,0

0,0,1,1,1,0,0,0,,0,0,0,0,0,0,1,0

1,1,0,0,1,1,X,X,,0,0,0,0,0,0,1,0

0,1,X,0,0,0,0,X,,0,0,0,0,0,0,1,0

1,0,1,X,0,0,X,1,,0,0,0,0,0,0,1,0

X,0,1,1,0,0,X,1,,0,0,0,0,0,0,1,0

1,X,0,X,0,1,1,1,,0,0,0,0,0,0,1,0

X,0,0,X,1,0,1,1,,0,0,0,0,0,0,1,0

1,X,1,0,X,1,0,1,,0,0,0,0,0,0,1,0

X,1,0,0,X,0,0,1,,0,0,0,0,0,0,1,0

X,1,1,1,0,1,X,0,,0,0,0,0,0,0,1,0

X,1,0,1,X,0,1,0,,0,0,0,0,0,0,1,0

X,1,1,0,0,X,0,0,,0,0,0,0,0,0,1,0

X,1,0,X,1,1,0,0,,0,0,0,0,0,0,1,0

0,1,0,0,0,1,1,X,,0,0,0,0,0,0,1,0

1,0,0,1,1,0,0,X,,0,0,0,0,0,0,1,0

0,0,1,X,0,1,1,1,,0,0,0,0,0,0,1,0

0,1,1,X,1,1,1,0,,0,0,0,0,0,0,1,0

1,1,0,0,1,0,X,X,,0,0,0,0,0,0,0,1

X,X,1,1,0,1,0,1,,0,0,0,0,0,0,0,1

1,1,X,1,1,X,0,0,,0,0,0,0,0,0,0,1

1,1,1,0,1,1,X,1,,0,0,0,0,0,0,0,1

1,X,1,0,0,1,0,0,,0,0,0,0,0,0,0,1

1,0,X,1,0,0,0,0,,0,0,0,0,0,0,0,1

X,X,0,1,1,1,0,X,,0,0,0,0,0,0,0,1

X,1,0,X,1,0,0,X,,0,0,0,0,0,0,0,1

1,1,0,1,0,1,X,X,,0,0,0,0,0,0,0,1

1,0,0,0,X,1,1,X,,0,0,0,0,0,0,0,1

X,1,1,1,1,0,X,1,,0,0,0,0,0,0,1,0

0,0,0,X,0,0,X,0,,0,0,0,0,0,0,1,0

1,1,X,1,X,1,0,0,,0,0,0,0,0,0,1,0

0,1,1,0,X,0,0,0,,0,0,0,0,0,0,1,0

0,X,0,0,X,0,X,1,,0,0,0,0,0,0,1,0

1,1,1,0,0,1,X,X,,0,0,0,0,0,0,1,0

0,1,0,0,0,X,X,1,,0,0,0,0,0,0,1,0

0,1,0,X,0,0,X,1,,0,0,0,0,0,0,1,0

X,1,1,1,X,1,1,1,,0,0,0,0,0,0,1,0

X,0,1,1,X,0,1,1,,0,0,0,0,0,0,1,0

1,X,0,1,0,X,0,1,,0,0,0,0,0,0,1,0

1,X,X,0,1,1,0,1,,0,0,0,0,0,0,1,0

1,1,0,0,1,X,X,0,,0,0,0,0,0,0,1,0

1,1,X,0,1,X,1,0,,0,0,0,0,0,0,1,0

0,X,X,1,0,0,1,0,,0,0,0,0,0,0,1,0

1,X,0,X,1,1,0,0,,0,0,0,0,0,0,1,0

1,X,X,0,0,1,0,0,,0,0,0,0,0,0,1,0

1,0,0,1,0,0,1,X,,0,0,0,0,0,0,1,0

0,1,0,X,1,1,1,1,,0,0,0,0,0,0,1,0

X,0,0,1,0,1,1,1,,0,0,0,0,0,0,1,0

X,0,1,0,1,0,1,0,,0,0,0,0,0,0,1,0

1,0,1,0,X,0,1,X,,0,0,0,0,0,0,0,1

0,X,1,0,1,1,X,0,,0,0,0,0,0,0,0,1

0,0,X,X,1,0,0,0,,0,0,0,0,0,0,0,1

0,0,0,1,X,1,0,0,,0,0,0,0,0,0,0,1

1,1,0,0,X,0,0,0,,0,0,0,0,0,0,0,1

0,0,1,1,0,0,1,1,,0,0,0,0,0,0,0,1

1,X,0,X,1,0,0,X,,0,0,0,0,0,0,0,1

0,0,0,1,1,1,X,X,,0,0,0,0,0,0,0,1

0,1,0,1,1,0,X,X,,0,0,0,0,0,0,0,1

1,1,1,X,1,1,1,X,,0,0,0,0,0,0,0,1

X,1,0,1,0,1,1,X„0,0,0,0,0,0,0,1
0,1,1,1,X,1,X,1„0,0,0,0,0,0,0,1
0,1,1,0,X,0,X,1„0,0,0,0,0,0,0,1
1,1,X,1,1,X,1,1„0,0,0,0,0,0,0,1
1,X,X,0,1,0,1,1„0,0,0,0,0,0,0,1
0,X,0,0,0,X,0,1„0,0,0,0,0,0,0,1
0,1,1,X,1,X,1,0„0,0,0,0,0,0,0,1
1,X,0,1,0,X,1,0„0,0,0,0,0,0,0,1
0,X,0,0,X,1,1,0„0,0,0,0,0,0,0,1
0,1,0,0,0,1,0,X„0,0,0,0,0,0,0,1
0,0,0,X,0,1,1,1„0,0,0,0,0,0,0,1
X,0,0,1,0,0,0,1„0,0,0,0,0,0,0,1
1,1,1,X,0,0,1,0„0,0,0,0,0,0,0,1
0,X,1,0,0,0,0,0„0,0,0,0,0,0,0,1
0,0,1,0,1,X,X,1„0,0,0,0,0,0,0,1
0,1,1,X,0,1,X,1„0,0,0,0,0,0,0,1
0,1,0,X,0,0,X,1„0,0,0,0,0,0,0,1
1,X,1,X,1,1,1,1„0,0,0,0,0,0,0,1
0,1,0,X,1,X,0,1„0,0,0,0,0,0,0,1
1,0,X,1,X,1,0,1„0,0,0,0,0,0,0,1
X,1,1,1,0,X,1,0„0,0,0,0,0,0,0,1
X,0,0,0,0,X,1,0„0,0,0,0,0,0,0,1
0,1,0,0,0,0,1,X„0,0,0,0,0,0,0,1
1,1,0,X,0,1,1,1„0,0,0,0,0,0,0,1
1,0,1,X,0,1,0,1„0,0,0,0,0,0,0,1
1,0,X,1,1,0,1,0„0,0,0,0,0,0,0,1
1,0,1,X,1,1,0,0„0,0,0,0,0,0,0,1

A.3 Мінімізована таблиця істинності третьої підстановки

x7,x6,x5,x4,x3,x2,x1,x0„s7,s6,s5,s4,s3,s2,s1,s0

1,X,1,1,0,1,X,1„1,0,0,0,0,0,0,0
0,0,0,0,X,0,X,0„1,0,0,0,0,0,0,0
1,X,1,1,1,X,0,0„1,0,0,0,0,0,0,0
0,0,0,1,1,0,X,1„1,0,0,0,0,0,0,0
0,0,X,X,0,X,1,1„1,0,0,0,0,0,0,0
1,1,0,X,1,1,1,X„1,0,0,0,0,0,0,0
1,0,X,0,1,1,0,X„1,0,0,0,0,0,0,0
X,1,1,0,1,0,0,X„1,0,0,0,0,0,0,0
0,1,0,1,0,X,X,1„1,0,0,0,0,0,0,0
0,X,1,0,0,1,X,1„1,0,0,0,0,0,0,0
1,X,0,1,0,0,X,1„1,0,0,0,0,0,0,0
0,X,0,1,X,0,1,1„1,0,0,0,0,0,0,0
X,0,X,0,1,1,0,1„1,0,0,0,0,0,0,0
X,1,1,1,X,0,1,1„1,0,0,0,0,0,0,0
0,1,0,X,1,X,0,0„1,0,0,0,0,0,0,0
0,1,1,1,1,0,1,X„1,0,0,0,0,0,0,0
0,0,1,1,0,0,X,0„1,0,0,0,0,0,0,0
0,0,0,1,0,1,X,X„1,0,0,0,0,0,0,0
1,0,X,0,0,1,1,X„1,0,0,0,0,0,0,0
1,0,0,0,X,0,0,X„1,0,0,0,0,0,0,0
1,0,1,0,1,X,X,1„1,0,0,0,0,0,0,0
1,1,0,0,0,X,X,1„1,0,0,0,0,0,0,0
X,0,1,0,1,0,X,1„1,0,0,0,0,0,0,0
X,0,0,1,1,X,1,1„1,0,0,0,0,0,0,0
1,0,1,1,X,X,0,1„1,0,0,0,0,0,0,0
0,1,X,0,X,0,0,1„1,0,0,0,0,0,0,0

X,1,1,0,X,0,0,1,,1,0,0,0,0,0,0,0
0,1,1,0,X,1,X,0,,1,0,0,0,0,0,0,0
0,0,1,X,X,1,1,0,,1,0,0,0,0,0,0,0
1,0,X,1,X,0,1,0,,1,0,0,0,0,0,0,0
0,0,0,X,X,1,0,0,,1,0,0,0,0,0,0,0
1,X,0,0,1,0,1,1,,1,0,0,0,0,0,0,0
1,0,0,0,0,X,1,0,,1,0,0,0,0,0,0,0
X,0,1,1,1,1,1,0,,1,0,0,0,0,0,0,0
1,X,1,0,1,0,1,0,,1,0,0,0,0,0,0,0
1,1,X,0,0,1,0,0,,1,0,0,0,0,0,0,0
X,1,X,1,X,1,1,0,,0,1,0,0,0,0,0,0
1,0,0,0,1,X,0,X,,0,1,0,0,0,0,0,0
1,X,0,X,1,0,1,1,,0,1,0,0,0,0,0,0
0,0,X,1,X,1,0,1,,0,1,0,0,0,0,0,0
X,0,0,0,X,0,0,1,,0,1,0,0,0,0,0,0
0,0,1,0,X,0,X,0,,0,1,0,0,0,0,0,0
1,1,X,X,0,0,0,0,,0,1,0,0,0,0,0,0
0,0,0,1,X,0,0,0,,0,1,0,0,0,0,0,0
0,0,1,0,1,X,1,X,,0,1,0,0,0,0,0,0
X,1,0,1,0,1,1,X,,0,1,0,0,0,0,0,0
1,1,1,0,0,X,0,X,,0,1,0,0,0,0,0,0
1,0,1,1,X,0,0,X,,0,1,0,0,0,0,0,0
1,1,X,0,X,0,1,1,,0,1,0,0,0,0,0,0
1,X,0,X,1,1,0,1,,0,1,0,0,0,0,0,0
1,0,X,X,0,0,0,1,,0,1,0,0,0,0,0,0
X,1,1,1,1,1,X,0,,0,1,0,0,0,0,0,0
X,1,0,1,0,X,1,0,,0,1,0,0,0,0,0,0
1,0,1,X,X,0,1,0,,0,1,0,0,0,0,0,0
X,1,1,0,0,X,0,0,,0,1,0,0,0,0,0,0
1,0,0,0,0,0,1,X,,0,1,0,0,0,0,0,0
1,0,1,1,0,1,X,1,,0,1,0,0,0,0,0,0

X,X,0,0,0,0,0,1,,1,0,0,0,0,0,0,0
0,1,X,0,0,X,1,0,,1,0,0,0,0,0,0,0
X,1,0,X,1,1,1,0,,1,0,0,0,0,0,0,0
0,X,X,0,0,0,1,0,,1,0,0,0,0,0,0,0
0,0,1,1,1,1,X,1,,1,0,0,0,0,0,0,0
0,1,1,1,1,X,0,1,,1,0,0,0,0,0,0,0
1,0,0,0,X,1,1,0,,1,0,0,0,0,0,0,0
1,1,X,0,1,0,1,0,,1,0,0,0,0,0,0,0
1,X,1,1,0,0,1,0,,1,0,0,0,0,0,0,0
1,X,1,0,0,0,0,0,,1,0,0,0,0,0,0,0
0,0,0,0,X,1,1,X,,0,1,0,0,0,0,0,0
0,0,X,1,X,0,1,1,,0,1,0,0,0,0,0,0
X,1,1,X,0,0,1,1,,0,1,0,0,0,0,0,0
0,X,X,0,1,1,0,1,,0,1,0,0,0,0,0,0
0,1,0,1,X,1,X,0,,0,1,0,0,0,0,0,0
1,X,X,1,1,1,1,0,,0,1,0,0,0,0,0,0
1,0,0,X,0,1,0,0,,0,1,0,0,0,0,0,0
1,0,1,1,1,X,1,X,,0,1,0,0,0,0,0,0
1,1,1,X,1,1,1,X,,0,1,0,0,0,0,0,0
0,0,1,X,0,0,1,X,,0,1,0,0,0,0,0,0
1,X,0,0,1,1,0,X,,0,1,0,0,0,0,0,0
X,1,1,0,0,X,1,1,,0,1,0,0,0,0,0,0
1,0,X,0,1,X,0,1,,0,1,0,0,0,0,0,0
0,X,0,X,1,0,0,1,,0,1,0,0,0,0,0,0
1,1,1,0,1,X,X,0,,0,1,0,0,0,0,0,0
0,1,0,0,X,X,1,0,,0,1,0,0,0,0,0,0
X,1,1,X,1,1,1,0,,0,1,0,0,0,0,0,0
0,1,X,1,1,X,0,0,,0,1,0,0,0,0,0,0
0,X,X,1,1,1,0,0,,0,1,0,0,0,0,0,0
0,1,1,1,0,1,X,1,,0,1,0,0,0,0,0,0
1,1,0,1,0,1,X,1,,0,1,0,0,0,0,0,0

0,1,1,1,1,0,X,1,,0,1,0,0,0,0,0,0

X,1,0,1,0,0,0,1,,0,1,0,0,0,0,0,0

0,0,1,X,0,1,0,0,,0,1,0,0,0,0,0,0

1,1,0,X,X,0,X,1,,0,0,1,0,0,0,0,0

1,0,1,0,1,1,X,X,,0,0,1,0,0,0,0,0

0,0,0,X,1,0,0,X,,0,0,1,0,0,0,0,0

X,0,1,1,0,X,0,1,,0,0,1,0,0,0,0,0

1,1,X,1,X,0,1,0,,0,0,1,0,0,0,0,0

0,1,1,0,0,X,1,1,,0,0,1,0,0,0,0,0

0,0,1,1,X,1,1,0,,0,0,1,0,0,0,0,0

0,X,1,0,1,0,1,0,,0,0,1,0,0,0,0,0

0,1,1,X,1,X,0,X,,0,0,1,0,0,0,0,0

X,1,0,X,1,X,0,1,,0,0,1,0,0,0,0,0

1,0,1,1,1,0,X,X,,0,0,1,0,0,0,0,0

1,X,0,1,0,1,1,X,,0,0,1,0,0,0,0,0

0,0,1,1,0,X,0,X,,0,0,1,0,0,0,0,0

1,X,0,0,0,1,0,X,,0,0,1,0,0,0,0,0

1,0,0,0,1,X,X,1,,0,0,1,0,0,0,0,0

X,0,1,1,0,0,X,1,,0,0,1,0,0,0,0,0

0,1,X,0,X,1,1,1,,0,0,1,0,0,0,0,0

1,1,X,1,1,1,X,0,,0,0,1,0,0,0,0,0

1,0,X,X,1,1,0,0,,0,0,1,0,0,0,0,0

1,0,X,1,0,0,1,1,,0,0,1,0,0,0,0,0

0,0,0,1,X,0,1,0,,0,0,1,0,0,0,0,0

1,X,0,1,0,0,0,0,,0,0,1,0,0,0,0,0

1,1,0,X,0,X,0,1,,0,0,0,1,0,0,0,0

0,0,X,0,1,0,X,0,,0,0,0,1,0,0,0,0

X,X,0,1,0,1,1,0,,0,0,0,1,0,0,0,0

0,X,0,0,X,1,0,0,,0,0,0,1,0,0,0,0

0,1,1,0,X,1,1,0,,0,0,0,1,0,0,0,0

0,X,0,0,0,0,1,0,,0,0,0,1,0,0,0,0

0,X,0,1,1,1,1,1,,0,1,0,0,0,0,0,0

0,0,0,X,1,0,1,0,,0,1,0,0,0,0,0,0

X,X,X,X,1,0,0,1,,0,0,1,0,0,0,0,0

X,0,1,X,X,0,0,1,,0,0,1,0,0,0,0,0

1,1,0,0,1,0,X,X,,0,0,1,0,0,0,0,0

0,0,0,1,X,1,X,1,,0,0,1,0,0,0,0,0

0,X,1,X,0,1,1,0,,0,0,1,0,0,0,0,0

0,1,0,0,0,0,0,X,,0,0,1,0,0,0,0,0

0,0,0,0,0,X,1,1,,0,0,1,0,0,0,0,0

X,1,0,1,1,1,1,0,,0,0,1,0,0,0,0,0

1,0,0,0,0,0,1,0,,0,0,1,0,0,0,0,0

1,0,1,X,X,0,0,X,,0,0,1,0,0,0,0,0

0,1,1,1,1,0,X,X,,0,0,1,0,0,0,0,0

0,0,X,1,0,1,1,X,,0,0,1,0,0,0,0,0

1,1,X,0,1,0,1,X,,0,0,1,0,0,0,0,0

0,X,1,0,1,1,0,X,,0,0,1,0,0,0,0,0

1,X,1,0,0,0,0,X,,0,0,1,0,0,0,0,0

0,X,1,0,1,1,X,1,,0,0,1,0,0,0,0,0

X,1,1,1,X,1,1,1,,0,0,1,0,0,0,0,0

0,1,X,1,X,1,0,1,,0,0,1,0,0,0,0,0

1,0,X,1,1,X,0,0,,0,0,1,0,0,0,0,0

1,X,1,0,0,1,1,1,,0,0,1,0,0,0,0,0

X,0,0,0,0,1,0,1,,0,0,1,0,0,0,0,0

1,1,1,X,0,1,0,0,,0,0,1,0,0,0,0,0

1,X,1,0,0,X,1,1,,0,0,0,1,0,0,0,0

X,X,1,1,1,0,0,1,,0,0,0,1,0,0,0,0

X,0,1,X,1,1,1,0,,0,0,0,1,0,0,0,0

1,1,1,X,X,1,0,0,,0,0,0,1,0,0,0,0

1,0,0,X,1,0,0,1,,0,0,0,1,0,0,0,0

1,1,0,X,0,1,1,0,,0,0,0,1,0,0,0,0

1,1,0,1,1,1,1,1,,0,0,0,1,0,0,0,0

0,X,1,X,1,1,1,X,,0,0,0,1,0,0,0,0

0,0,0,0,0,0,X,X,,0,0,0,1,0,0,0,0

0,0,1,X,0,1,0,X,,0,0,0,1,0,0,0,0

1,1,X,0,0,0,0,X,,0,0,0,1,0,0,0,0

0,1,0,1,X,0,X,1,,0,0,0,1,0,0,0,0

0,1,X,0,1,X,1,1,,0,0,0,1,0,0,0,0

X,1,1,X,0,0,1,1,,0,0,0,1,0,0,0,0

0,X,0,X,1,1,0,1,,0,0,0,1,0,0,0,0

1,0,1,1,1,X,X,0,,0,0,0,1,0,0,0,0

0,0,X,1,0,1,X,0,,0,0,0,1,0,0,0,0

1,X,1,0,0,0,X,0,,0,0,0,1,0,0,0,0

1,0,0,X,X,0,1,0,,0,0,0,1,0,0,0,0

1,0,X,1,X,1,0,0,,0,0,0,1,0,0,0,0

1,0,1,1,0,0,X,1,,0,0,0,1,0,0,0,0

0,0,0,1,0,X,0,1,,0,0,0,1,0,0,0,0

0,X,1,0,0,1,0,1,,0,0,0,1,0,0,0,0

0,1,1,1,0,X,0,0,,0,0,0,1,0,0,0,0

1,X,X,1,1,X,0,0,,0,0,0,0,1,0,0,0

X,0,X,X,1,1,0,0,,0,0,0,0,1,0,0,0

0,0,1,0,0,X,0,X,,0,0,0,0,1,0,0,0

X,X,0,0,0,1,1,1,,0,0,0,0,1,0,0,0

1,1,0,1,1,X,X,0,,0,0,0,0,1,0,0,0

1,1,0,1,X,0,X,0,,0,0,0,0,1,0,0,0

0,1,1,X,X,1,1,0,,0,0,0,0,1,0,0,0

0,1,0,1,0,1,0,X,,0,0,0,0,1,0,0,0

0,0,0,1,0,0,X,0,,0,0,0,0,1,0,0,0

0,1,0,1,1,1,1,1,,0,0,0,0,1,0,0,0

1,X,X,0,0,1,1,X,,0,0,0,0,1,0,0,0

1,0,X,X,0,0,X,1,,0,0,0,0,1,0,0,0

1,0,X,1,0,0,1,X,,0,0,0,0,1,0,0,0

0,0,0,0,1,X,0,X,,0,0,0,0,1,0,0,0

1,0,1,1,X,X,0,X,,0,0,0,1,0,0,0,0

X,0,0,1,1,0,1,X,,0,0,0,1,0,0,0,0

0,1,0,1,X,0,0,X,,0,0,0,1,0,0,0,0

0,1,1,0,1,X,X,1,,0,0,0,1,0,0,0,0

0,0,X,0,0,0,X,1,,0,0,0,1,0,0,0,0

0,X,1,1,X,1,1,1,,0,0,0,1,0,0,0,0

1,X,X,0,0,0,1,1,,0,0,0,1,0,0,0,0

X,0,X,1,1,1,0,1,,0,0,0,1,0,0,0,0

1,1,1,1,X,1,X,0,,0,0,0,1,0,0,0,0

1,1,0,X,1,0,X,0,,0,0,0,1,0,0,0,0

0,0,X,X,1,1,1,0,,0,0,0,1,0,0,0,0

0,1,0,X,1,X,0,0,,0,0,0,1,0,0,0,0

1,1,0,1,0,0,1,X,,0,0,0,1,0,0,0,0

0,0,0,0,X,1,1,1,,0,0,0,1,0,0,0,0

1,0,X,0,0,1,0,1,,0,0,0,1,0,0,0,0

0,1,X,1,1,0,1,0,,0,0,0,1,0,0,0,0

0,1,1,X,0,0,0,0,,0,0,0,1,0,0,0,0

1,X,X,X,1,1,0,0,,0,0,0,0,1,0,0,0

X,1,0,1,0,0,1,X,,0,0,0,0,1,0,0,0

1,X,1,0,0,0,X,1,,0,0,0,0,1,0,0,0

0,1,X,X,1,0,0,1,,0,0,0,0,1,0,0,0

1,0,1,1,X,1,X,0,,0,0,0,0,1,0,0,0

0,1,1,0,X,X,1,0,,0,0,0,0,1,0,0,0

1,X,1,1,X,1,0,0,,0,0,0,0,1,0,0,0

0,1,1,1,0,0,0,X,,0,0,0,0,1,0,0,0

0,0,0,1,0,X,1,0,,0,0,0,0,1,0,0,0

1,X,0,0,0,1,X,X,,0,0,0,0,1,0,0,0

1,0,1,X,X,0,X,1,,0,0,0,0,1,0,0,0

X,0,1,X,0,X,0,1,,0,0,0,0,1,0,0,0

0,0,1,1,1,X,0,X,,0,0,0,0,1,0,0,0

X,0,0,0,1,1,X,1,,0,0,0,0,1,0,0,0

1,1,X,1,X,0,1,1,,0,0,0,0,1,0,0,0

1,0,1,0,X,X,0,1,,0,0,0,0,1,0,0,0

1,0,0,X,X,1,0,1,,0,0,0,0,1,0,0,0

1,X,0,X,0,1,0,1,,0,0,0,0,1,0,0,0

X,0,X,0,1,0,1,0,,0,0,0,0,1,0,0,0

0,1,1,0,1,1,X,1,,0,0,0,0,1,0,0,0

1,X,0,0,1,0,1,1,,0,0,0,0,1,0,0,0

0,0,0,0,X,0,0,1,,0,0,0,0,1,0,0,0

0,1,1,0,1,0,X,0,,0,0,0,0,1,0,0,0

0,0,1,X,1,0,1,0,,0,0,0,0,1,0,0,0

1,1,0,X,0,0,1,0,,0,0,0,0,1,0,0,0

0,0,X,0,0,1,1,X,,0,0,0,0,0,1,0,0

1,X,0,0,0,1,0,X,,0,0,0,0,0,1,0,0

X,0,0,0,0,X,1,1,,0,0,0,0,0,1,0,0

0,0,1,X,0,0,0,1,,0,0,0,0,0,1,0,0

0,0,1,1,X,0,1,0,,0,0,0,0,0,1,0,0

0,1,X,1,0,X,1,X,,0,0,0,0,0,1,0,0

1,X,X,0,X,1,0,1,,0,0,0,0,0,1,0,0

1,1,1,X,1,1,1,X,,0,0,0,0,0,1,0,0

1,0,0,0,1,X,0,X,,0,0,0,0,0,1,0,0

0,X,0,0,1,1,0,X,,0,0,0,0,0,1,0,0

1,1,1,0,X,0,0,X,,0,0,0,0,0,1,0,0

1,0,0,X,0,1,X,1,,0,0,0,0,0,1,0,0

0,1,1,X,X,0,1,1,,0,0,0,0,0,1,0,0

X,1,0,0,X,0,0,1,,0,0,0,0,0,1,0,0

0,1,X,0,0,0,X,0,,0,0,0,0,0,1,0,0

1,X,1,1,X,1,1,0,,0,0,0,0,0,1,0,0

X,0,0,X,1,1,0,0,,0,0,0,0,0,1,0,0

0,X,X,1,0,1,0,0,,0,0,0,0,0,1,0,0

0,X,0,1,1,0,1,1,,0,0,0,0,0,1,0,0

0,X,0,1,1,1,1,0,,0,0,0,0,0,1,0,0

X,0,1,0,X,0,1,1,,0,0,0,0,0,1,0,0

0,X,0,0,1,X,0,1,,0,0,0,0,0,1,0,0

0,X,1,X,0,1,0,1,,0,0,0,0,0,1,0,0

0,X,1,0,0,1,X,0,,0,0,0,0,0,1,0,0

X,0,0,0,X,1,0,0,,0,0,0,0,0,1,0,0

X,1,1,1,0,1,1,1,,0,0,0,0,0,1,0,0

X,1,1,1,1,1,0,1,,0,0,0,0,0,1,0,0

0,1,0,0,1,1,X,0,,0,0,0,0,0,1,0,0

0,0,0,0,X,0,1,0,,0,0,0,0,0,1,0,0

0,1,0,X,1,0,1,0,,0,0,0,0,0,1,0,0

X,1,X,X,1,1,0,1,,0,0,0,0,0,1,0,0

1,0,1,0,X,0,1,X,,0,0,0,0,0,1,0,0

X,1,0,1,0,0,X,1,,0,0,0,0,0,1,0,0

1,0,1,X,1,0,1,1,,0,0,0,0,0,1,0,0

1,1,X,0,1,1,1,0,,0,0,0,0,0,1,0,0

1,X,0,1,1,0,1,0,,0,0,0,0,0,1,0,0

1,1,X,X,0,X,0,1,,0,0,0,0,0,1,0,0

0,X,1,1,0,X,X,0,,0,0,0,0,0,1,0,0

0,X,1,1,0,1,1,X,,0,0,0,0,0,1,0,0

X,0,1,1,1,1,0,X,,0,0,0,0,0,1,0,0

0,1,1,X,0,1,0,X,,0,0,0,0,0,1,0,0

X,1,1,1,1,0,0,X,,0,0,0,0,0,1,0,0

1,X,1,0,0,1,X,1,,0,0,0,0,0,1,0,0

X,0,0,1,0,X,0,1,,0,0,0,0,0,1,0,0

X,1,1,0,1,0,X,0,,0,0,0,0,0,1,0,0

0,1,X,1,X,1,1,0,,0,0,0,0,0,1,0,0

1,X,X,1,0,1,1,0,,0,0,0,0,0,1,0,0

0,X,X,0,1,1,0,0,,0,0,0,0,0,1,0,0

0,1,0,0,X,1,1,1,,0,0,0,0,0,1,0,0

0,0,0,1,X,0,0,1,,0,0,0,0,0,1,0,0

0,X,0,0,1,0,1,0,,0,0,0,0,0,1,0,0

X,0,0,0,1,0,1,0,,0,0,0,0,0,1,0,0

1,X,1,1,1,0,0,0,,0,0,0,0,0,1,0,0

0,X,0,0,1,0,X,X,,0,0,0,0,0,0,1,0

1,0,X,1,1,X,1,1,,0,0,0,0,0,0,1,0

0,0,X,0,1,0,X,0,,0,0,0,0,0,0,1,0

1,1,0,0,1,X,1,1,,0,0,0,0,0,0,1,0

0,0,X,0,0,1,0,1,,0,0,0,0,0,0,1,0

0,0,0,0,1,X,0,0,,0,0,0,0,0,0,1,0

X,X,0,0,1,0,0,X,,0,0,0,0,0,0,1,0

1,X,X,X,0,0,0,0,,0,0,0,0,0,0,1,0

1,1,0,X,1,0,0,X,,0,0,0,0,0,0,1,0

0,1,1,1,0,X,X,1,,0,0,0,0,0,0,1,0

1,1,X,1,0,1,X,1,,0,0,0,0,0,0,1,0

1,1,1,1,X,X,1,1,,0,0,0,0,0,0,1,0

X,1,0,X,0,0,1,1,,0,0,0,0,0,0,1,0

1,X,1,0,1,X,0,1,,0,0,0,0,0,0,1,0

1,0,X,1,X,1,0,1,,0,0,0,0,0,0,1,0

1,0,1,0,0,X,X,0,,0,0,0,0,0,0,1,0

X,1,1,0,0,0,X,0,,0,0,0,0,0,0,1,0

X,1,1,0,1,X,1,0,,0,0,0,0,0,0,1,0

0,1,X,X,1,1,1,0,,0,0,0,0,0,0,1,0

1,1,0,0,X,X,0,0,,0,0,0,0,0,0,1,0

1,1,1,0,0,1,1,X,,0,0,0,0,0,0,0,1,0

0,0,1,1,1,X,1,0,,0,0,0,0,0,0,0,1,0

1,0,X,X,0,0,0,X,,0,0,0,0,0,0,0,0,1

0,0,1,X,1,0,0,X,,0,0,0,0,0,0,0,0,1

0,1,X,1,1,0,X,1,,0,0,0,0,0,0,0,0,1

0,X,0,X,1,1,0,0,,0,0,0,0,0,0,0,0,1

0,0,1,1,X,0,1,1,,0,0,0,0,0,0,0,0,1

1,0,0,0,X,X,X,1,,0,0,0,0,0,0,0,0,1

1,0,0,0,1,0,X,X,,0,0,0,0,0,0,0,0,1

X,0,1,0,0,1,0,0,,0,0,0,0,0,0,1,0,0

1,0,0,X,0,0,0,0,,0,0,0,0,0,0,1,0,0

1,0,1,X,X,0,1,X,,0,0,0,0,0,0,0,1,0

X,0,1,1,1,X,0,1,,0,0,0,0,0,0,0,1,0

X,0,0,1,0,1,1,0,,0,0,0,0,0,0,0,1,0

0,1,1,1,X,0,0,0,,0,0,0,0,0,0,0,1,0

0,1,X,X,0,X,1,1,,0,0,0,0,0,0,0,1,0

1,X,1,0,1,1,0,X,,0,0,0,0,0,0,0,1,0

X,0,0,1,0,0,0,X,,0,0,0,0,0,0,0,1,0

0,1,1,1,X,1,X,1,,0,0,0,0,0,0,0,1,0

0,1,0,X,1,0,X,1,,0,0,0,0,0,0,0,1,0

0,X,0,1,0,X,1,1,,0,0,0,0,0,0,0,1,0

X,X,1,1,0,0,1,1,,0,0,0,0,0,0,0,1,0

1,1,X,0,0,X,0,1,,0,0,0,0,0,0,0,1,0

1,X,0,X,0,1,0,1,,0,0,0,0,0,0,0,1,0

1,0,1,1,X,0,X,0,,0,0,0,0,0,0,0,1,0

0,1,0,0,X,X,1,0,,0,0,0,0,0,0,0,1,0

0,0,X,1,X,1,1,0,,0,0,0,0,0,0,0,1,0

0,X,0,X,0,0,1,0,,0,0,0,0,0,0,0,1,0

X,0,1,1,0,X,0,0,,0,0,0,0,0,0,0,1,0

0,1,0,1,1,1,0,X,,0,0,0,0,0,0,0,1,0

1,X,0,1,1,1,0,0,,0,0,0,0,0,0,0,1,0

1,1,1,0,1,0,X,X,,0,0,0,0,0,0,0,0,1

0,1,1,1,1,X,X,1,,0,0,0,0,0,0,0,0,1

1,0,1,X,1,1,X,0,,0,0,0,0,0,0,0,0,1

0,0,1,0,0,1,X,1,,0,0,0,0,0,0,0,0,1

X,1,0,0,1,1,0,0,,0,0,0,0,0,0,0,0,1

X,X,0,1,0,X,1,0,,0,0,0,0,0,0,0,0,1

1,0,1,0,0,0,X,X,,0,0,0,0,0,0,0,0,1

0,X,0,0,1,1,1,X„0,0,0,0,0,0,0,1 1,1,0,X,0,1,1,X„0,0,0,0,0,0,0,1

0,1,1,0,X,0,1,X„0,0,0,0,0,0,0,1 X,1,0,1,0,1,0,X„0,0,0,0,0,0,0,1

1,1,0,1,X,0,0,X„0,0,0,0,0,0,0,1 1,X,0,1,1,1,X,1„0,0,0,0,0,0,0,1

1,X,1,1,1,X,1,1„0,0,0,0,0,0,0,1 X,X,0,0,0,0,1,1„0,0,0,0,0,0,0,1

1,0,X,1,0,X,0,1„0,0,0,0,0,0,0,1 1,X,1,0,X,0,0,1„0,0,0,0,0,0,0,1

0,1,X,X,0,0,0,1„0,0,0,0,0,0,0,1 0,X,0,X,0,0,0,1„0,0,0,0,0,0,0,1

0,1,0,0,0,X,X,0„0,0,0,0,0,0,0,1 0,0,X,0,1,1,X,0„0,0,0,0,0,0,0,1

X,0,X,1,0,1,1,0„0,0,0,0,0,0,0,1 0,X,X,0,0,1,1,0„0,0,0,0,0,0,0,1

0,1,1,X,X,0,1,0„0,0,0,0,0,0,0,1 X,0,0,X,1,0,1,0„0,0,0,0,0,0,0,1

1,0,X,0,0,X,0,0„0,0,0,0,0,0,0,1 X,0,X,0,0,0,0,0„0,0,0,0,0,0,0,1

1,0,1,1,0,1,X,1„0,0,0,0,0,0,0,1 1,1,1,0,0,1,X,1„0,0,0,0,0,0,0,1

0,0,0,1,X,1,1,1„0,0,0,0,0,0,0,1 1,1,X,1,0,0,1,1„0,0,0,0,0,0,0,1

0,1,0,0,1,X,0,1„0,0,0,0,0,0,0,1 1,1,X,1,1,0,1,0„0,0,0,0,0,0,0,1

1,X,1,0,0,0,1,0„0,0,0,0,0,0,0,1 1,1,1,X,0,1,0,0„0,0,0,0,0,0,0,1

A.4 Мінімізована таблиця істинності четвертої підстановки

x7,x6,x5,x4,x3,x2,x1,x0„s7,s6,s5,s4,s3,s2,s1,s0

1,1,X,1,X,0,1,X„1,0,0,0,0,0,0,0 1,1,1,X,1,0,X,1„1,0,0,0,0,0,0,0

1,0,X,1,0,0,X,1„1,0,0,0,0,0,0,0 1,0,0,X,0,X,1,1„1,0,0,0,0,0,0,0

0,X,0,X,1,0,1,1„1,0,0,0,0,0,0,0 0,0,1,1,X,X,1,0„1,0,0,0,0,0,0,0

1,1,0,X,X,1,1,0„1,0,0,0,0,0,0,0 1,X,1,0,0,X,0,0„1,0,0,0,0,0,0,0

0,X,X,0,1,0,0,0„1,0,0,0,0,0,0,0 0,1,1,1,0,0,0,X„1,0,0,0,0,0,0,0

1,0,X,1,1,0,0,0„1,0,0,0,0,0,0,0 0,0,0,0,0,0,0,1„1,0,0,0,0,0,0,0

0,0,0,0,0,0,1,0„1,0,0,0,0,0,0,0 1,X,0,1,1,1,X,X„1,0,0,0,0,0,0,0

1,X,0,1,0,0,X,X„1,0,0,0,0,0,0,0 1,X,0,1,X,X,1,0„1,0,0,0,0,0,0,0

0,0,0,1,1,X,1,X„1,0,0,0,0,0,0,0 0,X,1,1,0,1,1,X„1,0,0,0,0,0,0,0

0,1,X,0,1,0,1,X„1,0,0,0,0,0,0,0 X,1,0,1,0,0,1,X„1,0,0,0,0,0,0,0

1,1,0,1,0,X,0,X„1,0,0,0,0,0,0,0 0,0,1,1,X,1,X,1„1,0,0,0,0,0,0,0

0,1,0,0,X,1,X,1„1,0,0,0,0,0,0,0 0,X,0,1,1,0,X,1„1,0,0,0,0,0,0,0

1,1,X,0,1,0,X,1„1,0,0,0,0,0,0,0 X,0,X,1,0,1,1,1„1,0,0,0,0,0,0,0

1,X,1,X,1,0,1,1,,1,0,0,0,0,0,0,0

X,1,1,0,1,X,0,1,,1,0,0,0,0,0,0,0

1,0,0,X,X,1,0,1,,1,0,0,0,0,0,0,0

0,1,1,0,1,X,X,0,,1,0,0,0,0,0,0,0

1,1,0,X,0,0,X,0,,1,0,0,0,0,0,0,0

X,1,1,X,1,0,1,0,,1,0,0,0,0,0,0,0

X,1,0,0,X,0,0,0,,1,0,0,0,0,0,0,0

1,0,1,0,1,1,X,1,,1,0,0,0,0,0,0,0

0,1,0,1,1,X,0,1,,1,0,0,0,0,0,0,0

1,1,X,1,0,1,0,1,,1,0,0,0,0,0,0,0

1,0,1,0,X,0,1,0,,1,0,0,0,0,0,0,0

1,1,1,1,0,0,X,X,,0,1,0,0,0,0,0,0

1,0,X,0,1,0,1,X,,0,1,0,0,0,0,0,0

1,X,X,1,0,0,0,0,,0,1,0,0,0,0,0,0

X,1,0,1,0,0,1,1,,0,1,0,0,0,0,0,0

X,1,0,1,1,0,1,0,,0,1,0,0,0,0,0,0

0,0,0,1,1,0,1,1,,0,1,0,0,0,0,0,0

1,X,1,1,X,X,0,0,,0,1,0,0,0,0,0,0

1,X,1,0,0,0,1,X,,0,1,0,0,0,0,0,0

1,0,X,1,0,1,X,1,,0,1,0,0,0,0,0,0

X,0,1,0,1,X,1,1,,0,1,0,0,0,0,0,0

0,X,1,X,1,1,1,1,,0,1,0,0,0,0,0,0

0,X,X,1,1,1,0,1,,0,1,0,0,0,0,0,0

0,0,1,1,0,X,X,0,,0,1,0,0,0,0,0,0

0,1,0,X,0,1,X,0,,0,1,0,0,0,0,0,0

0,0,0,X,0,X,1,0,,0,1,0,0,0,0,0,0

X,0,0,X,0,1,1,0,,0,1,0,0,0,0,0,0

X,X,1,0,1,0,1,0,,0,1,0,0,0,0,0,0

X,1,1,X,0,0,0,0,,0,1,0,0,0,0,0,0

1,1,1,0,0,1,X,1,,0,1,0,0,0,0,0,0

0,1,1,0,0,0,X,1,,0,1,0,0,0,0,0,0

1,0,0,X,1,X,0,1,,1,0,0,0,0,0,0,0

0,1,1,X,X,1,0,1,,1,0,0,0,0,0,0,0

1,1,X,0,X,0,0,1,,1,0,0,0,0,0,0,0

0,1,0,X,0,1,X,0,,1,0,0,0,0,0,0,0

X,X,1,1,1,1,1,0,,1,0,0,0,0,0,0,0

X,0,1,0,X,1,0,0,,1,0,0,0,0,0,0,0

X,1,0,X,1,0,0,0,,1,0,0,0,0,0,0,0

0,1,1,0,0,X,1,1,,1,0,0,0,0,0,0,0

X,0,1,1,1,1,0,1,,1,0,0,0,0,0,0,0

0,0,1,0,0,1,X,0,,1,0,0,0,0,0,0,0

0,0,X,0,X,X,0,0,,0,1,0,0,0,0,0,0

0,0,1,1,0,0,X,X,,0,1,0,0,0,0,0,0

0,X,0,0,1,0,0,X,,0,1,0,0,0,0,0,0

0,1,1,1,X,1,1,1,,0,1,0,0,0,0,0,0

1,X,0,0,0,0,0,1,,0,1,0,0,0,0,0,0

1,1,X,0,1,1,0,0,,0,1,0,0,0,0,0,0

0,X,1,1,X,0,X,0,,0,1,0,0,0,0,0,0

1,0,1,X,1,1,1,X,,0,1,0,0,0,0,0,0

X,1,0,1,1,1,X,1,,0,1,0,0,0,0,0,0

X,1,1,1,1,X,1,1,,0,1,0,0,0,0,0,0

1,X,0,1,0,X,1,1,,0,1,0,0,0,0,0,0

0,1,0,X,X,1,0,1,,0,1,0,0,0,0,0,0

0,0,X,X,0,1,0,1,,0,1,0,0,0,0,0,0

1,1,1,1,X,1,X,0,,0,1,0,0,0,0,0,0

X,0,1,0,1,0,X,0,,0,1,0,0,0,0,0,0

X,0,X,1,1,1,1,0,,0,1,0,0,0,0,0,0

X,1,X,0,0,1,1,0,,0,1,0,0,0,0,0,0

X,0,X,0,0,1,0,0,,0,1,0,0,0,0,0,0

1,0,0,1,1,1,0,X,,0,1,0,0,0,0,0,0

1,0,1,1,1,0,X,1,,0,1,0,0,0,0,0,0

X,0,0,0,0,0,1,1,,0,1,0,0,0,0,0,0

1,0,1,0,X,1,0,1,,0,1,0,0,0,0,0,0

0,1,1,0,X,1,1,0,,0,1,0,0,0,0,0,0

0,1,0,0,X,0,1,0,,0,1,0,0,0,0,0,0

X,1,0,1,0,1,0,0,,0,1,0,0,0,0,0,0

X,1,1,0,X,X,1,1,,0,0,1,0,0,0,0,0

0,X,1,0,1,1,1,X,,0,0,1,0,0,0,0,0

0,1,0,0,0,X,0,X,,0,0,1,0,0,0,0,0

1,0,0,X,1,1,0,X,,0,0,1,0,0,0,0,0

1,1,0,X,1,0,X,1,,0,0,1,0,0,0,0,0

X,1,1,1,0,X,0,1,,0,0,1,0,0,0,0,0

X,0,1,0,X,0,0,1,,0,0,1,0,0,0,0,0

1,0,0,0,X,1,X,0,,0,0,1,0,0,0,0,0

0,0,1,0,1,0,0,X,,0,0,1,0,0,0,0,0

X,0,0,0,0,1,1,1,,0,0,1,0,0,0,0,0

0,X,0,1,0,0,1,1,,0,0,1,0,0,0,0,0

1,0,0,1,0,X,1,0,,0,0,1,0,0,0,0,0

1,0,0,1,1,X,0,0,,0,0,1,0,0,0,0,0

1,0,1,1,1,1,X,X,,0,0,1,0,0,0,0,0

0,X,0,1,1,1,1,X,,0,0,1,0,0,0,0,0

1,1,X,0,0,1,1,X,,0,0,1,0,0,0,0,0

1,X,0,1,1,1,0,X,,0,0,1,0,0,0,0,0

X,0,0,1,1,1,X,1,,0,0,1,0,0,0,0,0

0,1,1,X,0,1,X,1,,0,0,1,0,0,0,0,0

1,1,X,1,0,0,X,1,,0,0,1,0,0,0,0,0

1,0,X,0,0,0,X,1,,0,0,1,0,0,0,0,0

1,X,1,0,0,0,X,1,,0,0,1,0,0,0,0,0

0,1,0,1,X,X,1,1,,0,0,1,0,0,0,0,0

1,1,X,X,1,1,1,1,,0,0,1,0,0,0,0,0

1,1,1,0,X,1,X,0,,0,0,1,0,0,0,0,0

0,1,0,0,X,1,X,0,,0,0,1,0,0,0,0,0

0,X,1,1,1,1,X,0,,0,0,1,0,0,0,0,0

0,1,0,X,0,0,X,0,,0,0,1,0,0,0,0,0

1,X,1,1,1,X,1,0,,0,0,1,0,0,0,0,0

0,X,1,1,0,X,1,0,,0,0,1,0,0,0,0,0

1,X,1,0,X,0,1,0,,0,0,1,0,0,0,0,0

0,1,1,X,0,X,0,0,,0,0,1,0,0,0,0,0

X,0,0,1,X,1,0,0,,0,0,1,0,0,0,0,0

1,X,X,0,1,1,0,0,,0,0,1,0,0,0,0,0

0,X,X,0,0,1,0,0,,0,0,1,0,0,0,0,0

0,1,0,X,X,0,0,0,,0,0,1,0,0,0,0,0

X,1,1,1,X,0,0,0,,0,0,1,0,0,0,0,0

0,X,X,1,0,0,0,0,,0,0,1,0,0,0,0,0

X,X,0,0,0,0,0,0,,0,0,1,0,0,0,0,0

1,0,1,0,0,1,0,X,,0,0,1,0,0,0,0,0

0,1,1,1,1,0,X,1,,0,0,1,0,0,0,0,0

X,0,0,0,1,0,1,1,,0,0,1,0,0,0,0,0

0,0,0,X,1,0,1,0,,0,0,1,0,0,0,0,0

1,0,1,1,0,X,0,0,,0,0,1,0,0,0,0,0

1,1,0,1,0,X,0,0,,0,0,1,0,0,0,0,0

X,0,X,0,1,0,X,1,,0,0,0,1,0,0,0,0

1,0,0,X,0,0,1,X,,0,0,0,1,0,0,0,0

1,1,X,1,0,1,0,X,,0,0,0,1,0,0,0,0

0,1,0,X,0,0,0,X,,0,0,0,1,0,0,0,0

X,0,X,1,1,1,1,1,,0,0,0,1,0,0,0,0

X,0,0,1,X,0,1,1,,0,0,0,1,0,0,0,0

1,X,1,1,1,0,X,0,,0,0,0,1,0,0,0,0

1,0,X,0,0,0,X,0,,0,0,0,1,0,0,0,0

1,0,0,1,1,1,0,X,,0,0,0,1,0,0,0,0

1,1,0,0,1,0,X,0,,0,0,0,1,0,0,0,0

0,0,1,1,0,0,X,0,,0,0,0,1,0,0,0,0

0,1,0,1,1,1,0,0,,0,0,0,1,0,0,0,0

1,1,1,0,X,1,X,X,,0,0,0,1,0,0,0,0

0,0,X,0,1,X,0,X,,0,0,0,1,0,0,0,0

0,X,X,1,1,X,1,1,,0,0,0,1,0,0,0,0

0,X,0,0,1,1,1,X,,0,0,0,1,0,0,0,0

1,0,0,X,0,0,X,1,,0,0,0,1,0,0,0,0

1,1,1,0,X,X,0,1,,0,0,0,1,0,0,0,0

1,X,0,X,1,1,0,1,,0,0,0,1,0,0,0,0

X,1,0,1,X,0,0,1,,0,0,0,1,0,0,0,0

0,X,1,0,1,0,X,0,,0,0,0,1,0,0,0,0

0,X,1,0,X,1,1,0,,0,0,0,1,0,0,0,0

0,X,1,0,X,0,0,0,,0,0,0,1,0,0,0,0

1,1,0,1,1,0,1,X,,0,0,0,1,0,0,0,0

X,1,0,0,1,1,1,1,,0,0,0,1,0,0,0,0

X,1,1,0,0,0,1,1,,0,0,0,1,0,0,0,0

0,1,X,0,0,1,0,1,,0,0,0,1,0,0,0,0

0,0,0,1,1,0,X,0,,0,0,0,1,0,0,0,0

0,1,0,X,0,1,1,0,,0,0,0,1,0,0,0,0

1,X,0,0,0,0,1,0,,0,0,0,1,0,0,0,0

X,X,1,1,1,0,1,X,,0,0,0,0,1,0,0,0

0,1,0,X,0,0,1,X,,0,0,0,0,1,0,0,0

X,0,1,1,1,X,1,1,,0,0,0,0,1,0,0,0

1,0,0,X,X,0,0,1,,0,0,0,0,1,0,0,0

0,1,X,1,1,1,X,0,,0,0,0,0,1,0,0,0

0,X,1,X,0,1,1,0,,0,0,0,0,1,0,0,0

1,0,1,X,1,X,0,0,,0,0,0,0,1,0,0,0

1,0,1,1,X,0,1,1,,0,0,0,0,1,0,0,0

1,1,0,0,X,1,0,0,,0,0,0,0,1,0,0,0

1,0,X,1,1,0,1,X,,0,0,0,0,1,0,0,0

0,0,X,0,0,0,0,X,,0,0,0,0,1,0,0,0

0,0,0,0,X,1,X,1,,0,0,0,0,1,0,0,0

0,X,0,0,1,1,X,1,,0,0,0,0,1,0,0,0

X,1,1,0,0,1,X,1,,0,0,0,0,1,0,0,0

1,X,1,0,X,1,0,X,,0,0,0,1,0,0,0,0

X,X,1,0,X,1,0,1,,0,0,0,1,0,0,0,0

1,1,1,1,1,X,0,X,,0,0,0,1,0,0,0,0

1,1,X,1,0,X,1,1,,0,0,0,1,0,0,0,0

0,X,1,X,1,1,0,1,,0,0,0,1,0,0,0,0

X,0,0,X,1,1,0,1,,0,0,0,1,0,0,0,0

0,1,1,0,0,X,X,0,,0,0,0,1,0,0,0,0

0,1,X,0,1,X,1,0,,0,0,0,1,0,0,0,0

0,0,1,X,1,X,0,0,,0,0,0,1,0,0,0,0

0,1,0,1,0,1,1,X,,0,0,0,1,0,0,0,0

1,1,1,1,0,0,1,X,,0,0,0,1,0,0,0,0

0,0,1,X,0,1,1,1,,0,0,0,1,0,0,0,0

1,0,1,1,0,X,0,1,,0,0,0,1,0,0,0,0

0,X,1,1,0,0,0,1,,0,0,0,1,0,0,0,0

0,0,1,0,0,X,X,0,,0,0,0,0,1,0,0,0

X,1,1,0,X,1,1,0,,0,0,0,0,1,0,0,0

0,1,X,0,1,0,0,X,,0,0,0,0,1,0,0,0

0,X,0,X,0,0,1,1,,0,0,0,0,1,0,0,0

0,0,1,0,0,X,X,0,,0,0,0,0,1,0,0,0

X,1,1,0,X,1,1,0,,0,0,0,0,1,0,0,0

1,0,1,1,X,X,0,0,,0,0,0,0,1,0,0,0

0,1,1,1,0,0,0,X,,0,0,0,0,1,0,0,0

1,0,0,1,0,1,X,0,,0,0,0,0,1,0,0,0

0,1,1,1,1,1,X,X,,0,0,0,0,1,0,0,0

0,X,0,1,0,1,0,X,,0,0,0,0,1,0,0,0

1,1,0,1,X,1,X,1,,0,0,0,0,1,0,0,0

0,0,0,X,1,1,X,1,,0,0,0,0,1,0,0,0

X,0,0,0,1,1,X,1,,0,0,0,0,1,0,0,0

0,X,0,0,0,0,X,1,,0,0,0,0,1,0,0,0

1,1,0,0,X,X,1,1,,0,0,0,0,1,0,0,0

X,1,1,X,1,1,0,1,,0,0,0,0,1,0,0,0

0,1,1,0,1,X,X,0,,0,0,0,0,1,0,0,0

0,X,0,0,0,X,1,0,,0,0,0,0,1,0,0,0

1,1,X,X,0,1,0,0,,0,0,0,0,1,0,0,0

1,0,0,1,1,1,0,X,,0,0,0,0,1,0,0,0

0,0,X,1,0,1,0,1,,0,0,0,0,1,0,0,0

1,1,1,X,0,0,0,1,,0,0,0,0,1,0,0,0

1,1,0,X,1,0,1,0,,0,0,0,0,1,0,0,0

X,0,1,1,1,1,0,0,,0,0,0,0,1,0,0,0

X,0,0,1,X,0,1,X,,0,0,0,0,0,1,0,0

0,1,1,X,0,1,0,X,,0,0,0,0,0,1,0,0

0,1,X,1,0,0,X,1,,0,0,0,0,0,1,0,0

X,0,1,X,1,0,1,0,,0,0,0,0,0,1,0,0

1,0,1,1,1,0,X,1,,0,0,0,0,0,1,0,0

0,1,1,1,X,0,1,1,,0,0,0,0,0,1,0,0

1,X,0,0,1,1,1,0,,0,0,0,0,0,1,0,0

0,0,0,1,X,1,0,0,,0,0,0,0,0,1,0,0

0,0,0,1,1,X,1,X,,0,0,0,0,0,1,0,0

1,1,X,1,0,1,1,X,,0,0,0,0,0,1,0,0

0,1,0,X,0,0,1,X,,0,0,0,0,0,1,0,0

0,0,1,0,1,X,X,1,,0,0,0,0,0,1,0,0

0,0,X,X,1,1,1,1,,0,0,0,0,0,1,0,0

X,1,X,0,0,0,1,1,,0,0,0,0,0,1,0,0

1,X,0,0,0,X,0,1,,0,0,0,0,0,1,0,0

0,X,1,X,0,0,0,1,,0,0,0,0,0,1,0,0

0,X,1,0,0,X,1,0,,0,0,0,0,0,1,0,0

1,X,0,0,X,0,0,0,,0,0,0,0,0,1,0,0

1,1,1,1,0,0,0,X,,0,0,0,0,0,1,0,0

0,1,0,0,1,0,X,1,,0,0,0,0,0,1,0,0

1,X,1,0,1,1,1,1,,0,0,0,0,0,1,0,0

0,1,1,X,0,X,1,1,,0,0,0,0,0,1,0,0

0,0,1,X,X,0,0,1,,0,0,0,0,0,1,0,0

1,1,1,X,0,X,1,0,,0,0,0,0,0,1,0,0

0,0,X,0,X,1,1,0,,0,0,0,0,0,1,0,0

1,X,1,X,1,0,0,0,,0,0,0,0,0,1,0,0

1,0,0,0,0,1,0,X,,0,0,0,0,0,1,0,0

0,0,X,1,1,0,0,1,,0,0,0,0,0,1,0,0

1,X,1,0,0,0,0,1,,0,0,0,0,0,1,0,0

1,X,1,0,0,0,0,0,,1,0,0,0,0,0,1,0

1,0,0,X,1,1,0,1,,0,0,0,0,0,1,0,0

0,1,1,X,0,X,1,1,,0,0,0,0,0,1,0,0

0,0,1,X,X,0,0,1,,0,0,0,0,0,1,0,0

1,1,1,X,0,X,1,0,,0,0,0,0,0,1,0,0

0,0,X,0,X,1,1,0,,0,0,0,0,0,1,0,0

1,X,1,X,1,0,0,0,,0,0,0,0,0,1,0,0

1,0,0,0,0,1,0,X,,0,0,0,0,0,1,0,0

0,0,X,1,1,0,0,1,,0,0,0,0,0,1,0,0

1,X,1,0,0,0,0,1,,0,0,0,0,0,1,0,0

1,X,1,0,0,0,0,0,,1,0,0,0,0,0,1,0

1,0,0,X,1,1,0,1,,0,0,0,0,0,1,0,0

1,0,0,X,0,0,0,1,,0,0,0,0,0,1,0,0

0,X,0,0,1,0,1,0,,0,0,0,0,0,1,0,0

0,1,0,0,0,X,0,0,,0,0,0,0,0,1,0,0

0,X,1,1,1,0,0,0,,0,0,0,0,0,1,0,0

1,1,0,X,0,0,0,0,,0,0,0,0,0,1,0,0

X,1,0,X,0,0,1,X,,0,0,0,0,0,0,1,0

X,X,0,1,0,X,1,0,,0,0,0,0,0,0,1,0

0,0,1,0,X,0,X,1,,0,0,0,0,0,0,1,0

1,0,0,1,1,1,0,X,,0,0,0,0,0,0,1,0

1,X,0,1,1,1,1,1,,0,0,0,0,0,0,1,0

1,0,X,1,0,0,1,1,,0,0,0,0,0,0,1,0

1,1,0,1,X,0,1,0,,0,0,0,0,0,0,1,0

1,1,1,1,0,1,1,1,,0,0,0,0,0,0,1,0

0,0,X,0,1,X,X,1,,0,0,0,0,0,0,1,0

0,1,1,0,0,1,X,X,,0,0,0,0,0,0,1,0

X,1,0,0,1,1,0,X,,0,0,0,0,0,0,1,0

X,0,0,1,0,0,0,X,,0,0,0,0,0,0,1,0

0,0,0,X,0,1,X,1,,0,0,0,0,0,0,1,0

X,1,0,0,0,X,1,1,,0,0,0,0,0,0,1,0

X,0,0,0,0,1,X,0,,0,0,0,0,0,0,1,0

0,0,X,1,1,0,X,0,,0,0,0,0,0,0,1,0

X,0,X,0,0,1,1,0,,0,0,0,0,0,0,1,0

0,X,0,0,X,0,1,0,,0,0,0,0,0,0,1,0

0,X,1,X,1,0,0,0,,0,0,0,0,0,0,1,0

0,0,1,1,1,1,0,X,,0,0,0,0,0,0,1,0

1,0,0,0,1,X,1,0,,0,0,0,0,0,0,1,0

1,1,1,1,1,X,0,0,,0,0,0,0,0,0,1,0

X,1,1,1,1,1,X,1,,0,0,0,0,0,0,0,1

1,0,X,0,1,X,1,1,,0,0,0,0,0,0,0,1

1,X,0,X,0,0,1,1,,0,0,0,0,0,0,0,1

0,1,X,1,1,0,X,0,,0,0,0,0,0,0,0,1

1,1,1,1,0,X,1,0,,0,0,0,0,0,1,0,0

0,0,0,0,1,X,0,0,,0,0,0,0,0,1,0,0

1,0,0,X,1,0,0,0,,0,0,0,0,0,1,0,0

X,1,1,1,1,0,0,0,,0,0,0,0,0,1,0,0

1,1,X,0,0,0,X,X,,0,0,0,0,0,0,1,0

1,1,0,X,0,X,X,0,,0,0,0,0,0,0,1,0

1,X,1,0,1,1,1,X,,0,0,0,0,0,0,1,0

0,X,0,X,1,0,0,1,,0,0,0,0,0,0,1,0

0,X,1,1,1,1,1,1,,0,0,0,0,0,0,1,0

0,X,1,0,1,0,1,1,,0,0,0,0,0,0,1,0

0,1,1,X,0,1,0,1,,0,0,0,0,0,0,1,0

1,0,1,X,0,1,0,0,,0,0,0,0,0,0,1,0

1,1,0,0,X,X,0,X,,0,0,0,0,0,0,1,0

1,0,0,X,X,X,0,1,,0,0,0,0,0,0,1,0

1,0,1,X,1,1,1,X,,0,0,0,0,0,0,1,0

X,1,1,0,0,1,0,X,,0,0,0,0,0,0,1,0

1,X,1,0,0,0,0,X,,0,0,0,0,0,0,1,0

1,X,0,0,1,0,X,1,,0,0,0,0,0,0,1,0

1,0,X,X,1,0,0,1,,0,0,0,0,0,0,1,0

1,1,1,0,X,0,X,0,,0,0,0,0,0,0,1,0

X,0,1,1,1,0,X,0,,0,0,0,0,0,0,1,0

0,X,1,1,X,0,1,0,,0,0,0,0,0,0,1,0

0,1,1,X,X,0,0,0,,0,0,0,0,0,0,1,0

0,0,1,1,1,0,1,X,,0,0,0,0,0,0,1,0

1,X,1,1,1,1,0,1,,0,0,0,0,0,0,1,0

0,1,X,0,1,1,1,0,,0,0,0,0,0,0,1,0

0,0,X,1,1,1,1,X,,0,0,0,0,0,0,0,1

1,1,0,0,X,X,1,1,,0,0,0,0,0,0,0,1

1,0,X,0,X,0,1,1,,0,0,0,0,0,0,0,1

X,0,X,0,0,1,0,1,,0,0,0,0,0,0,0,1

1,1,1,0,X,X,1,0,,0,0,0,0,0,0,0,1

X,X,0,1,0,1,1,0,,0,0,0,0,0,0,0,1 1,X,1,0,X,0,0,0,,0,0,0,0,0,0,0,1

1,0,1,1,1,0,1,X,,0,0,0,0,0,0,0,1 0,0,1,0,0,1,X,1,,0,0,0,0,0,0,0,1

1,1,X,1,0,1,0,1,,0,0,0,0,0,0,0,1 0,0,1,0,X,0,1,0,,0,0,0,0,0,0,0,1

1,X,1,X,1,0,X,1,,0,0,0,0,0,0,0,1 0,X,0,X,1,X,1,1,,0,0,0,0,0,0,0,1

X,0,0,X,X,0,1,1,,0,0,0,0,0,0,0,1 0,1,0,0,1,X,0,X,,0,0,0,0,0,0,0,1

1,0,0,0,X,1,0,X,,0,0,0,0,0,0,0,1 X,0,0,1,1,1,0,X,,0,0,0,0,0,0,0,1

1,0,0,1,X,0,X,1,,0,0,0,0,0,0,0,1 0,0,0,1,X,X,1,1,,0,0,0,0,0,0,0,1

X,1,0,X,1,1,1,1,,0,0,0,0,0,0,0,1 1,1,1,X,X,0,0,1,,0,0,0,0,0,0,0,1

1,1,X,X,1,0,0,1,,0,0,0,0,0,0,0,1 0,0,X,X,0,0,0,1,,0,0,0,0,0,0,0,1

0,0,0,1,1,X,X,0,,0,0,0,0,0,0,0,1 1,0,0,0,1,X,X,0,,0,0,0,0,0,0,0,1

0,1,X,0,0,1,X,0,,0,0,0,0,0,0,0,1 1,1,X,0,0,0,X,0,,0,0,0,0,0,0,0,1

X,0,0,X,1,1,1,0,,0,0,0,0,0,0,0,1 1,0,X,X,0,1,1,0,,0,0,0,0,0,0,0,1

1,1,X,1,1,X,0,0,,0,0,0,0,0,0,0,1 0,1,X,0,1,X,0,0,,0,0,0,0,0,0,0,1

1,X,1,1,X,1,0,0,,0,0,0,0,0,0,0,1 X,0,0,X,0,1,0,0,,0,0,0,0,0,0,0,1

0,X,X,0,0,1,0,0,,0,0,0,0,0,0,0,1 X,1,0,X,0,0,0,0,,0,0,0,0,0,0,0,1

0,0,1,1,0,0,1,X,,0,0,0,0,0,0,0,1 0,1,1,0,0,0,1,X,,0,0,0,0,0,0,0,1

1,0,0,1,0,0,0,X,,0,0,0,0,0,0,0,1 1,X,1,1,0,1,1,1,,0,0,0,0,0,0,0,1

0,1,1,0,X,0,1,1,,0,0,0,0,0,0,0,1 X,0,1,0,1,1,0,0,,0,0,0,0,0,0,0,1